Software Engineering Institute

Models for Evaluating and Improving Architecture Competence

Len Bass
Paul Clements
Rick Kazman
Mark Klein

March 2008

TECHNICAL REPORT
CMU/SEI-2008-TR-006
ESC-TR-2008-006

Software Architecture Technology Initiative

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Table of Contents

Acknowledgments **vii**

Abstract **ix**

1 Introduction **1**
1.1 Terminology and Definitions 2
1.2 Models of Competence 7
1.3 Organization of This Report 9

2 The Duties, Skills, and Knowledge (DSK) Model **11**
2.1 What Are an Architect's Duties, Skills, and Knowledge? 12
2.2 Advantages and Challenges of the Approach 13
2.3 Processing the Raw Data 15
2.4 Duties 16
2.5 Skills 17
2.6 Knowledge 18
2.7 Using the DSK Model to Assess and Improve the Architecture Competence of Individuals 21
2.8 Duties, Skills, and Knowledge for a Software Architecture Organization 22

3 The Human Performance Technology Model **25**
3.1 Using the Human Performance Technology Model to Measure and Improve Architecture Competence 27

4 The Organizational Coordination Model **29**
4.1 Dependency 29
4.2 The Coordination Capability of an Organization 30
4.3 Measuring the Coordination Activities 31
4.4 Relating Organizational Capability to Dependencies 32

5 The Organizational Learning Model **33**
5.1 The Components of the Organizational Learning Framework 34
5.2 Using the Organizational Learning Framework to Measure and Improve Architecture Competence 35

6 Considering the Models Together **37**
6.1 How the Models Together Support Evaluation 37
6.2 Principles Embodied by the Models 38
6.3 Coverage Provided by the Models 39

7 Building an Assessment Instrument **43**
7.1 Assessment Outcomes 43
7.2 The Foundations and Structure of the Instrument 44
7.3 Sample Questions 45
7.4 Reflections on the Instrument Questions 47

8 Summary **49**
8.1 Next Steps 49
8.2 Conclusion 51

Appendix A: Survey of Practicing Architects **53**

Appendix B: Complete List of Duties, Skills, and Knowledge **61**

Bibliography **69**

List of Figures

Figure 1: Teodorescu and Binder's Components and Processes for Building a Competence Model 4

Figure 2: Software Engineering Artifacts, Transformations, and Verifications 7

Figure 3: Skills and Knowledge Support the Execution of Duties 12

Figure 4: Architecture Job Accomplishments 27

Figure 5: Gilbert's Behavior Engineering Model 28

Figure 6: Dependencies Between Modules with Coordination Between Development Teams 29

Figure 7: Factors that Affect Coordination Capability of an Organization 31

Figure 8: The Relationship Between Architecture Design Competence and Organizational Learning 33

Figure 9: "Acquiring Architects" in One Kind of Organization Interact with "Producing Architects" in Another 50

List of Tables

Table 1:	The Duties of a Software Architect	16
Table 2:	The Skills of a Software Architect	18
Table 3:	Body of Knowledge Needed by a Software Architect	20
Table 4:	The Knowledge Areas of a Software Architect	21
Table 5:	Priority Each Model Assigns to Observing Artifacts, Process, and Organization	40
Table 6:	Applicability of the Models to Individuals, Teams, and Organizations	40
Table 7:	Duties of a Software Architect	61
Table 8:	Skills of a Software Architect	65
Table 9:	Knowledge of a Software Architect	66

Acknowledgments

Our thanks go to many people who have helped with this work:

- Participants in the birds-of-a-feather session at the 2006 Working IFIP/IEEE Conference on Software Architecture (WICSA) laid the groundwork for the research, and expressed interest and encouragement.
- Divya Devesh carried out the web searches to compile much of the material about courses and certificate programs that appear in the appendices. Prageti Verma, Shivani Reddy, and Divya Devesh carried out the research to capture an architect's duties, skills, and knowledge. Prageti Verma summarized the duties submitted by visitors to the SEI Software Architecture website.
- The IFIP working group WG2.10 on software architecture listened to our explanation of the approach and offered suggestions. Rich Hilliard was especially helpful. Mary Shaw provided comments and pointed us to the results of the education working session at WICSA. Philippe Kruchten provided many pointers, engaged in helpful discussions, and contributed to an earlier version of this report.
- Tommi Mikkonen (University of Tampere) provided excellent clarifying comments. Joe Batman (SEI) graciously allowed us to use his piece on organizations and architecture. John Klein (Avaya) suggested new survey questions and new organizational practices (process improvement, measuring quality of past architectures). Poornachandra Sarang (ABCOM Information Systems Pvt. Ltd., Mumbai, India), Prof. TV Prabhakar (IIT-Kanpur), and David Weiss (Avaya) all made insightful suggestions about the work. Hans van Vliet provided helpful comments about our questionnaire. Steve Miller (dean, School of Information Systems, Singapore Management University) participated in a fruitful discussion leading to a figure in this report that describes the different organizational roles and shows how architecture competence is different in each (Figure 9: "Acquiring Architects" in One Kind of Organization Interact with "Producing Architects" in Another).
- Members of the SEI Software Architecture Technology Initiative listened to progress reports and provided feedback.
- Joe Batman, Suzanne Garcia, Linda Northrop, and Mark Staples of the SEI provided insightful reviews that helped us improve the report.

Abstract

Software architecture competence is the ability of an individual or organization to acquire, use, and sustain the skills and knowledge necessary to carry out software architecture-centric practices. Previous work in architecture has concentrated on its technical aspects: methods and tools for creating, analyzing, and using architecture. However, a different perspective recognizes that these activities are carried out by people working in organizations, and those people and organizations can use assistance towards consistently producing high-quality architectures.

This report lays out the basic concepts of software architecture competence and describes four models for explaining, measuring, and improving the architecture competence of an individual or a software-producing organization. The models are based on (1) the duties, skills, and knowledge required of a software architect or architecture organization, (2) human performance technology, an engineering approach applied to improving the competence of individuals, (3) organizational coordination, the study of how people and units in an organization share information, and (4) organizational learning, an approach to how organizations acquire, internalize, and utilize knowledge to improve their performance. The report also shows how the four models can be synergistically applied to produce an evaluation instrument to measure an organization's architecture competence.

1 Introduction

Software architecture has become recognized in recent years as an indispensable part of the development process for high-quality, software-intensive systems. With a few notable exceptions, the field of software architecture has been primarily devoted to technical and technological aspects of architecture, including but not limited to

- architecture-based development methodologies
- architectural design solutions involving styles, patterns, tactics, and other cataloged solutions or solution fragments
- evaluating, analyzing, or assessing a software architecture for suitability or fitness of purpose
- capturing and communicating a software architecture using languages, notations, templates, and tools
- modeling of systems based on their architectural descriptions
- the relationship between software architecture and implementation—either taking the architecture to code or recovering the software architecture from a legacy code base
- architecture-based technology platforms, infrastructures, layers, frameworks, and prepackaged components that currently dominate the open market, and the standards associated with them
- the relationship between software architecture and testing

These topics and others form the backbone of a formidable body of technical work [Shaw 2006]. However, none of them is focused on the **software architects** who work within organizations to create, evaluate, maintain, and promulgate software architectures. Only if people and organizations are equipped to effectively carry out software-architecture-centric practices will organizations routinely produce high-quality architectures that are aligned with their business goals. An organization's ability to do this well cannot be understood simply through examination of past architectures and measurement of their deficiencies. The root causes of those deficiencies need to be understood. Therefore the goal of our competence work is this:

> *We wish to be able to effectively measure the competence of software architects, software architect teams, and software-architecture-producing organizations and to prescribe effective ways in which competence can be improved.*

The purpose of this report is to identify human and organizational factors that are critical to achieving the full promise of software architecture.

1.1 TERMINOLOGY AND DEFINITIONS

Before delving into the factors that contribute to architecture competence, we will explore definitions of competence and related terms, discuss their implications, and propose our definition of architecture competence.

Architect, architecture: Throughout this report, *architect* and *architecture* should be taken to mean "software architect" and "software architecture," respectively, unless otherwise qualified. As is usually the case, we expect that many of the concepts related to software architecture apply equally well to system architecture or enterprise architecture. However, the scope of this report is, for now, limited to software.

Competence: There are several nontechnical definitions of *competence*; the following are typical:

> *Competence: the quality of being competent; adequacy; possession of required skill, knowledge, qualification, or capacity [Webster 1996]*
>
> *Competent: having suitable or sufficient skill, knowledge, experience, etc., for some purpose; properly qualified [Random House 2006]*
>
> *Competent: Capable of performing an allotted or required function [American Heritage 2002]*

These definitions reveal a predisposition towards measuring qualities of the individual: skill, knowledge, qualification, experience, or capability. This contrasts sharply with the definition given by Gilbert:

> *Competent people are those who can create valuable results without using excessively costly behavior [Gilbert 1996, p.17].*

This definition scrupulously avoids measuring the individual and instead measures the result. Gilbert's model of competence will be explored in Section 3. We will develop a precise definition of competence in architecture as we go along. For now, we simply want to call attention to the dichotomy between definitions of competence that target individual workers and definitions that target the results of their work.

Organizational competence: Much of the literature in competence deals with competence of organizations rather than individuals. For example, Taatila deals explicitly with organizational competence in a comprehensive fashion. Taatila writes that *organizational competence* refers to

> *an organization's internal capability to reach stakeholder-specific situation-dependent goals, where the capability consists of the situation-specific combination of all of the possible individual-based, structure-based, and asset-based attributes directly manageable by an organization and available to the organization in the situation [Taatila 2004, p. 4]*

Briefly, organizational competence measures those internal attributes that enable an organization to reach its targets. Taatila is quick to point out that organizational competence is fundamentally affected by the competence of individuals employed by that organization. Attributes that individuals contribute to their organization include their

- creativity
- intelligence
- knowledge and skills
- behavioral traits (including such aspects as honesty and maturity)
- motivation
- commitment
- communication capabilities

Taatila identifies competence-related attributes of an organization, including

- how roles are assigned to employees
- guiding principles
- defined organizational processes
- organizational culture (including values, atmosphere, and practices)
- organizational knowledge
- managerial practices
- organizational learning
- information and information technology systems
- work environment

Finally, Taatila writes that the assets that an organization holds—for example, products and production environments—affect its ability to meet its target goals.

Teodorescu and Binder prescribe a way to build a competence model that can be used to measure and increase an organization's progress towards achieving competence [Teodorescu 2004]. Their model is shown in Figure 1. Important inputs include the goals of the organization. Processes are built to identify and confirm the goals, conduct performance analyses, and so forth. Then remedial actions are planned and implemented as needed.

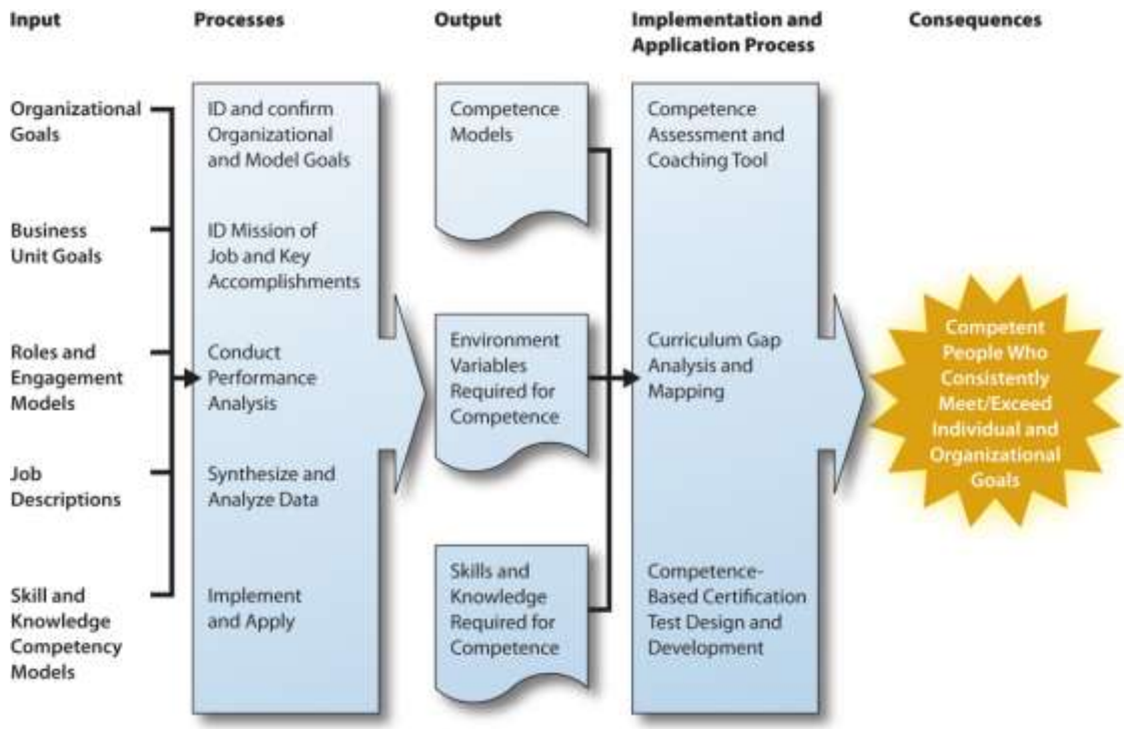

Figure 1: Teodorescu and Binder's Components and Processes for Building a Competence Model [Teodorescu 2004, p. 10]

Competency: When studying competence, a related term often arises: namely, *competency*. For example, Woodruffe defines competency as follows:

> *A competency is the set of behaviour patterns that the incumbent needs to bring to a position in order to perform its tasks and functions with competence" [Woodruffe 1993, p. 29].*

Similarly, Michael Lewis writes that

> *Organisational competencies are those combinations of organisational resource and process that together underpin sustainable competitive advantage for a specific firm competing in a particular product/service market [Lewis 2001, p. 190].*

A competency can be a core competency for an organization, a concept about which the management literature contains a wealth of information. For example, Prahalad and Hamel claim that a focus on core competencies distinguishes more successful from less successful companies:

> *Core competence does not diminish with use. Unlike physical assets, which do deteriorate over time, competencies are enhanced as they are applied and shared. But competencies still need to be nurtured and protected; knowledge fades if it is not used. Competencies are the glue that binds existing businesses… Consider 3M's competence with sticky tape. In dreaming businesses as diverse as "Post-it" notes, magnetic tape, photographic film, pressure-sensitive tapes, and coated abrasives, the company has brought to bear widely shared competencies in substrates, coatings, and adhesives and devised various ways to combine them.*

Indeed, 3M has invested consistently in them. What seems to be an extremely diversified portfolio of businesses belies a few shared competencies [Hamel 1990, p. 82].

For a grand tour of the various meanings of *competency* to be found in the literature, as well as a summary of the term's evolution over time, see Hoffman's treatment [Hoffman 1999].

Overall, *competency* refers to an ability or a capability, usually in an organization, whereas *competence* refers to how well a capability is exercised. The distinction is subtle, and not critical to this report. We will explore software architecture **competency** by articulating the specifics of the capability and how that capability can be carried out **with competence** by individuals and organizations.

Individual vs. organizational competence: We have been speaking of individual and organizational competence as though they were independent. There are cases where individual architects are prevented from producing high-quality architectures or from performing duties satisfactorily by the encompassing organization. For example, the organization may rush an architecture out of its creation stage and into coding before sufficient validation has been done. Is the individual who created that architecture competent (but the organization not competent) when the organization's failure to perform its duties effectively leads to a failed architecture? Alternately, the most architecturally-invested organization will produce only failures if its architects are not competent. Suppose an organization competently carries out its duties, such as adequately funding the architecture stage of projects, insuring high-quality reviews, paying its architects well, facilitating information exchange among its architects, and so forth. If its architects do not perform their duties competently, is the organization competent? Does the organization's failure to hire and retain competent architects mean it is incompetent?

Individual and organizational competencies are intertwined. Studying only one or the other will not do. This idea reinforces our position that simply examining completed architectures and measuring their deficiencies will not do. We need to understand the root causes of those deficiencies.

In fact, the competence of an organization is intertwined with the competence of external organizations with which it interacts—suppliers, customers, regulators, and so forth. An incompetent supplier can ruin a system just as surely as incompetent architects—if the organization fails to guard against it. While this demonstrates that the web of competence is potentially unlimited, we have chosen for reasons of practicality to keep our scope focused on the competence of the individual architect and his or her employing organization.

Architecture competence: What do we mean by architecture competence? Summarizing the previous treatments of competence, we can take one of two tracks. Using the Gilbert sense of the term in which we measure the output rather than the qualifications of the actor, we can posit the following:

A competent software architect is one who produces high-quality software architectures with acceptable cost. An organization competent in software architecture is one that consistently produces high-quality software architecture with acceptable cost.

Taking the more conventional point of view, we can posit a definition of *architecture competence* in line with the first definition we cited, dealing with the "possession of required skill, knowledge, qualification, or capacity." Thus

> *Architecture competence is the ability of an individual, team, or organization to effectively carry out the functions and activities necessary to produce architectures that are aligned with the developing organization's business goals.*

The former concentrates on past results; the latter concentrates on current abilities. Both have their advantages and disadvantages. The first definition

- fails to take into account the fact that successful architects do more than produce architectures. Architects evaluate other architectures, mentor apprentices, work with management, communicate with stakeholders, consult with developers, are technological visionaries, and perform a host of other activities that we need to include under the umbrella of competence. These ancillary activities must be included because organizations expect them, as indicated by a large sample of position descriptions for software architects gathered as part of the research for this work. If our research results in prescribing a path to competence that leads to an architect's being regarded as deficient by an employer, it will be a disservice to the profession. It can be argued that all of these activities are geared towards producing high-quality architectures in the future, and hence fall naturally into our definition. We simply must not be short sighted in what it means to "produce" an architecture.
- requires an architect to present results before his or her competence can be evaluated. A newly appointed architect, by this definition, cannot have any competence at all!
- assumes that an architect's past performance is a good predictor of future performance, discounting any factors in past organizational environments (possibly in a completely different organization) or technological environments (possibly using technology that is now obsolete) that might have enhanced or inhibited his or her ability to perform, or makes the assumption that the architect can easily adapt to new environments and demands
- assumes that an architect's past efforts are available and measurable

On the other hand, the second definition

- takes on faith the fact that present qualifications are good predictors of future performance
- assumes that we know the right "functions and activities" to measure that will predict the ability of an individual or organization to produce high-quality architectures in the future

It seems inevitable that any holistic approach to architecture competence will include aspects of past performance as well as present environment and activities.

At this point we can posit a definition of *architecture competence* for an organization that reflects both present activities and past results, and accounts for the competence of individuals, teams, and organizations:

The architecture competence of an organization is the ability of that organization to grow, use, and sustain the skills and knowledge necessary to effectively carry out architecture-centric practices at the individual, team, and organizational levels to produce architectures with acceptable cost that lead to systems aligned with the organization's business goals.

This will be the working definition of competence for this report, although we eventually hope to craft a more concise one.

1.2 MODELS OF COMPETENCE

Figure 2 shows how software architecture is positioned among certain other key software engineering artifacts and activities. The solid arcs above the circles show transformation from one stage to another. Light dashed arcs below the circles show verification.[1] Along the bottom, the figure also shows some of the artifacts that emerge from the stages, such as a specification of the organization's business goals [Kazman 2005], the requirements for the system being developed, and the architecture.

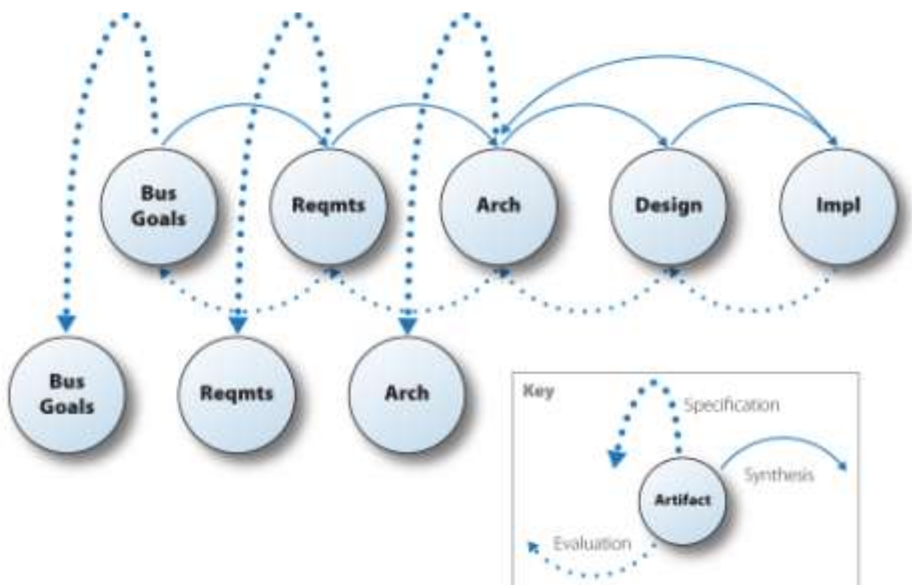

Figure 2: Software Engineering Artifacts, Transformations, and Verifications

Figure 2 is not intended to be a definitive architecture-based life-cycle model—every organization is likely to follow its own whether captured in a diagram or not—but rather a sketch showing the most important activities to which architecture contributes and by which architecture is informed.

The architecture-centric practices mentioned in our definition of competence can be seen as those practices that (a) allow an organization or individual to traverse the arcs in this diagram or

1 Testing is treated in this figure as a validation activity from implementation to requirements.

(b) allow an organization or individual to improve the execution of those practices across systems and over time.

Our research has uncovered four distinct models of organizational and human behavior that can be applied to software architecture and help us evaluate and improve how individuals and organizations traverse the arcs in Figure 2. These models are

1. Duties, Skills, and Knowledge (DSK) model of competence: This model is predicated on the belief that architects and architecture-producing organizations are useful sources for understanding the tasks necessary to the job of architecting. Developing this model will involve cataloging what architects and organizations do and know, building measures for how well they do and know it, and crafting improvement strategies for their duties, skills, and knowledge.
2. Human Performance model of competence: This model is based on the human performance engineering work of Thomas Gilbert [Gilbert 1996]. This model is predicated on the belief that competent individuals in any profession are the ones who produce the most valuable results at a reasonable cost. Developing this model will involve figuring out how to measure the worth and cost of the outputs of architecture efforts, finding areas where that ratio can be improved, and crafting improvement strategies based on environmental and behavioral factors.
3. Organizational Coordination model of competence: This model is being developed through ongoing research related to multisite development of software. The focus is on creating an inter-team coordination model for teams developing a single product or a closely related set of products. The architecture for the product induces a requirement for teams to coordinate during the realization or refinement of various architectural decisions. The organizational structure, practices, and tool environment of the teams allow for particular types of coordination with a particular inter-team communication bandwidth. The coordination model of competence will compare the requirements for coordination that the architecture induces with the bandwidth for coordination supported by the organizational structure, practices, and tool environment [Cataldo 2007].
4. Organizational Learning model of competence: This model is based on the concept that organizations, and not just individuals, can learn. Organizational learning is a change in the organization that occurs as a function of experience. This change can occur in the organization's cognitions or knowledge (e.g., as presented by Fiol and Lyles [Fiol 1985]), its routines or practices (e.g., as demonstrated by Levitt and March [Levitt 1988]), or its performance (e.g., as presented by Dutton and Thomas [Dutton 1984]). Although individuals are the medium through which organizational learning generally occurs, learning by individuals within the organization does not necessarily imply that organizational learning has occurred. For learning to be organizational, it has to have a supra-individual component [Levitt 1988]. To measure organizational learning, we can consider three approaches: (1) measure knowledge directly though questionnaires, interviews, and verbal protocols; (2) treat changes in routines and practices as indicators of changes in knowledge; or (3) view changes in organizational performance indicators associated with experience as reflecting changes in knowledge [Argote 2007].

We will use and integrate the best parts of each of these models to create a useful evaluation instrument for architecture competence, as well as to identify specific improvement strategies.

1.3 ORGANIZATION OF THIS REPORT

The remainder of this report is organized as follows. Sections 2, 3, 4, and 5 address the DSK, Human Performance Technology, Organizational Coordination, and Organizational Learning models, respectively.[2] Section 6 considers the four models as a group and makes observations about their synergy. Section 7 explains how we will build an evaluation instrument based on the four models. Section 8 summarizes the report, describes related work and its relevance, and lays out future work in this area.

[2] Section 2 details the Duties, Skills, and Knowledge model. Its treatment is the longest of the four. The other three models exist as a body of separate work in the open literature, but DSK models were inventoried explicitly as part of our research in architecture competence, and this report is the definitive summary of that research.

2 The Duties, Skills, and Knowledge (DSK) Model

The ideal architect should be a man of letters, a skillful draftsman, a mathematician, familiar with historical studies, a diligent student of philosophy, acquainted with music, not ignorant of medicine, learned in the responses of jurisconsults, familiar with astronomy *and astronomical calculations.*

– *Vitruvius, De Architectura (25 BC)*

Expertise in German / French / Japanese… will be an added advantage.

– *from a position description for Senior Technical Architect at Infosys Technologies Ltd. in India, 2006*

One of the two main schools of thought presented in Section 1 holds that competence deals with the skills and knowledge necessary to carry out assigned tasks. Someone who is competent is someone who is "capable of performing an allotted or required function." For the purpose of this report, the tasks are those of software architects. To help architects improve, we need to first understand what they do. What are their specific duties? What skills and knowledge make them "capable of performing their allotted or required function?"

Architects perform many activities beyond directly producing an architecture. These activities, which we call duties, form the backbone of individual architecture competence. A survey of the broad body of information aimed at architects (such as websites, courses, books, and position descriptions for architects) as well as a survey of practicing architects tell us that duties are but one aspect. Writers about architects also speak of skills and knowledge. For example, the ability to communicate ideas clearly is a skill often ascribed to competent architects. Courses aimed at architects imply that architects need to have up-to-date knowledge about topics such as patterns, database platforms, web services standards, or quality attributes.

Therefore, duties, skills, and knowledge[3] form a triad upon which architecture competence for individuals rests. We hypothesize that the relationship among these three is as shown in Figure 3—namely, skills and knowledge support the ability to perform the required duties.[4] Omniscient, infinitely talented architects are of no use if they cannot (for whatever reason) perform the duties required of the position; we might say that such people possess great potential, but we would not say they were competent.

3 Some writers speak of the importance of experience. We catalog experience as a form of knowledge.

4 Figure 3 glosses over other relationships that are present. For example, the **skill** of abstract thinking is informed by **knowledge** of abstractions that have been previously discovered and characterized.

To give examples of these concepts, “design the architecture” is a duty, “ability to think abstractly” is a skill, and “patterns, styles, and tactics” is a part of the body of knowledge. This example purposely illustrates that skills and knowledge are important (only) for supporting the ability to carry out duties effectively. As another example, “documenting the architecture” is a duty, “ability to write clearly” is a skill, and “ANSI/IEEE Std. 1471/2000” is part of the related body of knowledge. Of course, a skill or knowledge area can support more than one duty.

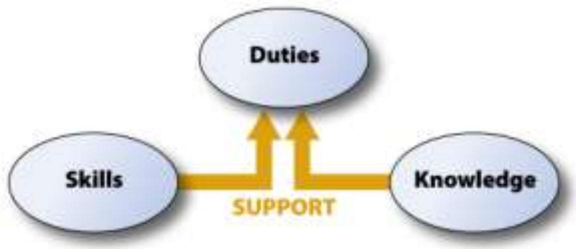

Figure 3: Skills and Knowledge Support the Execution of Duties

2.1 WHAT ARE AN ARCHITECT’S DUTIES, SKILLS, AND KNOWLEDGE?

Assembling a comprehensive set of duties, skills, and knowledge for architects can help us define what it means to be a competent architect. To assemble this set we surveyed approximately 200 sources of information targeted to professional architects in the summer of 2006. The results of this survey show what those sources describe as the key duties, skills, and knowledge of the trade. We present a distillation—a categorization—of the results of the survey [Clements 2007].

Although there is no single definitive or authoritative source for the duties, skills, and knowledge required for competence in architecture, there are several community resources that we have canvassed to assemble a picture of what an architect and an architecting organization must know and do. We divided our information sources into three categories:

1. **Broadcast sources** are sources of information written by self-styled experts aimed at mass anonymous consumption. These sources include
   - websites related to software architecture. We performed a web search for sites describing or giving advice on software architecture. There are many sites and portals on software architecture. Well-known examples include Bredemeyer’s site and the Carnegie Mellon® Software Engineering Institute (SEI) architecture website [Bredemeyer 2007, SEI 2008]. For several years, the SEI site has provided a forum for architects to contribute lists of their most important architectural duties. We took data from the 16 websites we found that explicitly mentioned duties, skills, or knowledge (although we visited many more).
   - blogs and essays related to software architecture. “Things to Do in Denver If You’re an Architect,” by van Ommering, is a good example of an essay that explicitly discusses architectural duties, skills, and knowledge [van Ommering 2005]. In all, we took data from 16 essays.

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

- books on software architecture. By looking at the detailed tables of contents (that www.amazon.com makes available online) of the best-selling books on software architecture, we can infer what authors are prescribing that architects do, have, and know. The 25 best-selling titles were surveyed; we made inferences from 23 of them.

2. **Sources of training and education** tell us what organizations in the business of education or training think that architects need to know and what architects (or aspiring architects) are paying money to learn. These sources include
   - university courses in software architecture. An extensive web search revealed 29 academic courses in software architecture. The course descriptions and syllabi provided lists of the duties, skills, and knowledge for architects being taught in these courses.
   - public nonacademic (industrial) courses in software architecture. We gathered data from 22 industrial courses whose descriptions were available online.
   - certificate and certification programs for software architects.[5] We identified and gathered data from seven programs.
3. **Sources related to "doing architecture for a living"** tell us what employers are looking for and what architects seeking employment are saying about themselves. This category turned out to be an especially rich source of duties, skills, and knowledge that were often listed and described in exactly those terms. These sources include
   - job descriptions for software architects. We visited the websites of the top 150 Fortune 500 companies and searched for position descriptions for software architects. We also visited major employment sites. We gathered information from 60 job descriptions.
   - résumés of software architects seeking employment. We harvested about a dozen résumés from employment sites.

We are currently distributing a questionnaire to practicing software architects and will add this fourth source to the corpus in the future.

2.2 ADVANTAGES AND CHALLENGES OF THE APPROACH

The practice of surveying the community to arrive at a picture of best practices has several things that recommend it. The first is that such surveying avoids definition wars. We do not dwell on defining what an architect is or does, and what architecture is; for these things, we simply accept the weight of evidence provided by community consensus.

Second, surveying does not limit data by assuming that only a particular career path leads to becoming an architect. For the most part, surveying also makes unnecessary the discussion of different kinds of positions—senior designer, software architect, chief architect, software solution architect, IT architect, and so forth. Our focus includes anyone who produces software architectures.

---

5 A *certificate* indicates successful completion of a course of study. A *certification* indicates tested mastery of information or abilities.

Third, we believe that the approach works equally well for individual and organizational competence in architecture. Organizations have architecture-centric duties (e.g., establishing an architecture review board or giving adequate schedule and budget to a project's architecture activities), skills (e.g., human resource skills for adequately hiring, mentoring, and rewarding architects), and knowledge (e.g., how to assemble the most effective architecture teams). Our data gathering for organizational duties, skills, and knowledge is still in progress and will not be addressed in this report.

Fourth, the approach is systematic and removes us from the need to address competence in an ad hoc fashion. There are only so many sources of knowledge, and it is in fact feasible to gather a representative sample of each kind. Further, the sources are not limited to one industry, geographic region, or economic sector.

Fifth, focusing on duties, skills, and knowledge provides an operational way to assess current competence (measure the effectiveness of performance of the architect's duties, the strength of the skills, and the extent of the knowledge) as well as to predict future competence (measure the skills and the mastery of the knowledge). It also suggests an obvious and actionable approach to improve individual competence: practice the duties, improve your skills, and master the knowledge.

The approach had its challenges. The foremost was deciding whether a given data source was truly referring to a software architect. A bewildering variety of current job titles contain the word *architect.* "IT architect," "solution architect," "software systems architect," "enterprise architect," "Java architect," "middleware architect," "platform architect," and "enterprise architect" are just a few examples. We even encountered "code architect."

Our approach was to automatically accept data about any title that contained the words *software* and *architect*. We also decided to include "IT architect" and other roles we found on the basis of the software-intensive material we observed in those job descriptions. We explicitly decided not to include enterprise architects, since enterprise architecture is related to, but different from, the profession we are targeting. We then looked at the remaining job descriptions on a case-by-case basis. If the information explicitly mentioned duties, skills, or knowledge as applied to software architecture, we included it. For books we were stricter—the title had to include the words *software* and either *architect* or *architecture*.

The second challenge was dealing with data (e.g., a position description) that was declared to be for a software architect but was clearly written using the word only as a prestigious title for a developer. Again, we handled this on a case-by-case basis, looking into the duties, skills, and knowledge mentioned by the source for something actually related to architecture.

A third challenge, which turned out to be less vexing than we anticipated, was assigning data to categories. For example, is leadership a duty, a skill, or a kind of knowledge? What about mentoring? Here, we let the information source guide us. Where "leadership" was written as something the architect had to do or perform, it became a duty. If it was written as something the architect

had to be good at, we listed it as a skill. If it was written as something the architect had to know about or know how to do, it went into the knowledge bin.

The fourth challenge was knowing when to stop surveying. Where it was practical, we carried out exhaustive surveys of particular sources. For example, we cataloged every university software architecture course that was returned in Google searches. In cases where exhaustive searches were not practical, we stopped when it seemed that subsequent sources were not revealing any new information. This was a subjective assessment.

Fifth, what to do with like-sounding entries? For example, is "gather requirements" the same duty as "interact with stakeholders to see what their needs are?" Is "accommodating" the same skill as "flexible?" What shall we do with "document the software" and "document the software using views meaningful to the stakeholders?" There were hundreds of conundrums like this, which we finessed by avoiding the problem completely. Instead of merging the data as we collected it, we took the approach of treating each piece of advice as a legitimate and unique contribution. Only in the case of identical or near-identical wording did we merge two items into one (but then we counted that one as occurring twice). We then engaged an affinity diagramming exercise to produce clusters, which we have used as the basis of our results. The affinity diagramming exercise is explained in the Section 2.3.

2.3 PROCESSING THE RAW DATA

Our information gathering resulted in over 400 duties, skills, and knowledge areas, each of which **somebody** thinks is important for software architects to master. The guidance we found for architects ranges from the broad and predictable…

- "analyze and validate the architecture"
- perform "tradeoff analysis"
- "prepare architectural documents and presentations"

…to the prescriptively methodical….

- "choose the set of views that will be most valuable to the architecture's community of stakeholders"
- "measure results using quantitative metrics and improve both personal results and teams' productivity"

…to the ethereal bordering on spiritual…

- have "political sagacity"
- "focus on the big picture"
- "build trusted advisor relationships"
- "know yourself"

Viewing all of it—the mundane and the inspiring, the obvious and the unexpected, the popular and the esoteric—as a single body of work resulted in an emerging picture of what "the community" (as defined by the sources we polled) believes are the attributes we should ascribe to a software architect. The result was about 200 separately cataloged duties, about 100 skills, and about 100 areas of knowledge. To extract order from the chaos, we performed an affinity diagram exercise to add structure to the duties, skills, and knowledge.

The affinity diagram was originally developed by anthropologist Kawakita to aid in discovering meaningful groups of ideas from a raw list [Tague 2005]. Kawakita's approach is to examine the list and let groupings emerge naturally and intuitively, rather than by following a pre-ordained categorization [Beyer 1998]. An affinity diagram allows for categories that are not mutually exclusive. The steps of the affinity diagram process are (briefly) as follows:

1. Assemble the team.
2. Write individual statements on note cards.
3. Group the statements.
4. Name each group.
5. Cluster the groups.

In our case, we performed three separate affinity exercises (one each for duties, skills, and knowledge), which took about eight hours over three consecutive days. Our affinity exercise did not result in the allocation of any datum to more than one cluster, although unique membership was not a constraint of the exercise.

2.4 DUTIES

This section summarizes the sources we found that speak to an architect's duties. The data and the results of the affinity exercise are shown in Table 1.

Table 1: The Duties of a Software Architect

General Duty Area	Specific Duty Area
Architecting	Creating an architecture
	Architecture evaluation and analysis
	Documentation
	Existing system and transformation
	Other architecting duties not specific to the above categories
Life-cycle phases other than architecture	Requirements
	Coding
	Testing

Table 1: The Duties of an Architect, cont'd.

General Duty Area	Specific Duty Area
Life-cycle phases other than architecture (cont'd.)	Future technologies
	Tools and technology selection
Interacting with stakeholders	Interacting with stakeholders in general, or stakeholders other than clients or developers
	Clients
	Developers
Management	Project management
	People management
	Support for management
Organization and business related	Organization
	Business
Leadership and team building	Technical leadership
	Team building

2.5 SKILLS

Given the wide range of duties enumerated in the previous section, what skills (beyond mastery of the technical[6] body of knowledge) does an architect need to possess? Much has been written about the architect's special role of leadership in a project; the role requires the architect to be an effective communicator, manager, team builder, visionary, and mentor.

Some certificate or certification programs emphasize nontechnical skills; for example, Microsoft offers certification programs for Infrastructure Architects and Solutions Architects. Common to both certification programs are nontechnical assessment areas of leadership, organization dynamics, and communication [Microsoft 2008].

Architecture consultant Dana Bredemeyer has written extensively about the skill set needed by an architect [Bredemeyer 2007]. His website lists a number of nontechnical skills necessary for a software architect. The August 2005 newsletter of the International Association of Software Architects, *Perspectives of the IASA*, includes an article titled "System Architect: Necessity, Not Luxury" by Jeffcoat and Yaghoobi [Jeffcoat 2005]. They write that in addition to fulfilling technical responsibilities, the architect must also serve as leader, mentor, liaison, designer, and visionary (and the authors provide a paragraph about each).

6 We use *technical* in this report to describe information related to computer science or software engineering, and *nontechnical* to refer to other kinds of skills or knowledge.

The UK Chapter of the International Council on Systems Engineering (INCOSE) maintains a "Core Competencies Framework" for systems engineers that includes a "Basic Skills and Behaviours" section listing "the usual common attributes required by any professional engineer" [INCOSE 2005]. The list includes coaching, communicating, negotiating, influencing, and "team working."

As a final note, Turley and Bieman identify a set of 38 "competencies" for software engineers that we would call *skills* [Turley 1995]. We mention this only in passing, as our work concentrates on skills for software architecting, a specialization of software engineering. There is overlap, of course, but the skill sets are not identical.

Table 2 is a full set of skills gleaned from our information gathering.

Table 2: The Skills of a Software Architect

General Skill Area	Specific Skill Area
Communication skills	External communication skills
	Communication skills in general
	Internal communication skills
Interpersonal skills	Within team
	Outside of team
Work skills	Leadership
	For effectively managing workload
	For excelling in corporate environment
	For handling information
Personal skills	Personal qualities
	For handling unknown factors
	For handling unexpected developments
	Learning skills

2.6 KNOWLEDGE

A competent architect has an intimate familiarity with the architectural body of knowledge. The software architect should

- be comfortable with all branches of software engineering from requirements definition to implementation, development, and verification and validation
- be familiar with supporting disciplines such as configuration management and project management
- understand current design and implementation technologies

Knowledge and experience in one or more application domains is also necessary.

The body of knowledge issue was addressed at the 2006 Working International Federation for Information Processing/Institute of Electrical and Electronics Engineers (IFIP/IEEE) Conference on Software Architecture (WICSA). A working group categorized the knowledge needed by a software architect [WICSA 2007]. Table 3 presents a summary of this categorization. The numbers in the cells are derived from Bloom's taxonomy for categorizing levels of abstraction [Bloom 2004]. The dual column headings reflect the dual purpose of the table: It was constructed to guide the building of an academic curriculum as well as to help practicing architects in career planning. Headings describe experience level and type of degree held: Bachelor of Science (BSc) or Master of Science (MSc) in Computer Science (CS) or Software Engineering (SE).

Table 3: Body of Knowledge Needed by a Software Architect

Topic	**CS undergrad / practitioner with BSc in CS**	**New programmer / practitioner with MSc in CS**	**New architect / practitioner with MSc in SE with focus on SA or with significant experience**
Basic software engineering knowledge: programming, decomposition, version control, and so forth	1	3	5
People knowledge: leadership, teamwork, communication, negotiation, accepting direction, mentoring, consulting, and so forth	0	3	3-5
Business knowledge	0	1	3
Architecture techniques: large-scale synthesis, complexity management (abstraction, decomposition, etc.), synthesis, analysis, patterns, evaluation, and so forth	1	2	4
Requirements engineering	1	1-2	4
Software project management: deployment, process, estimation, and so forth	1	1	3 or more
Programming	2	4	2
Platform technology: databases, networks, embedded, enterprise, integration tools	2	4	2
Systems engineering	–	–	–
Architecture documentation	0	1	5
Reuse and integration	1	2	4-5
Domain knowledge	1	1	5
Mentoring	–	–	–

Key to table entries: 1=knowledge; 2=comprehension; 3=application; 4=analysis; 5=synthesis; 6=evaluation

Although these categories provide a well-considered starting point, there is no agreed-upon body of technical knowledge that a practicing software architect must master. However, we can construct an overview by surveying a number of sources such as public courses, certification programs, online job descriptions, and best-selling software architecture books.

Table 4 is a full set of knowledge areas gleaned from our information gathering.

Table 4: The Knowledge Areas of a Software Architect

General Knowledge Area	Specific Knowledge Area
Computer science knowledge	Knowledge of architecture concepts
	Knowledge of software engineering
	Design knowledge
	Programming knowledge
Knowledge of technologies and platforms	Specific technologies and platforms
	General knowledge of technologies and platforms
Knowledge about the organization's context and management	Domain knowledge
	Industry knowledge
	Enterprise knowledge
	Leadership and management techniques and experience

2.7 USING THE DSK MODEL TO ASSESS AND IMPROVE THE ARCHITECTURE COMPETENCE OF INDIVIDUALS

In the DSK model, the key to competence lies in the duties, skills, and knowledge of an architect. The greater the architect's ability to carry out the duties and possess the required skills and knowledge, the more able that architect is to produce high-quality architectures and hence the more competent. We can assess architecture competence according to the DSK model (and our other models, as we will show in Section 7). The results of such an assessment instrument can be used to identify areas where needed improvements are indicated.

Given the duties, skills, and knowledge cataloged in our survey, what avenues are available to an individual software architect to *improve* his or her competence? Three strategies emerge from the DSK model:

1. **Gain experience carrying out the duties:** Apprenticeship is a productive path to achieving experience. Education alone is not enough, because education without on-the-job application merely enhances knowledge.
2. **Improve nontechnical skills:** This dimension of improvement involves taking professional development courses, for example, in leadership or time management.
3. **Master the body of knowledge:** One of the most important things a competent architect must do is master the body of knowledge and remain up-to-date on it. Taking courses, becoming certified, reading books and journals, visiting websites and portals, reading blogs, attending architecture-oriented conferences, joining a professional societies, and meeting with other architects are all useful ways to improve knowledge.

2.8 DUTIES, SKILLS, AND KNOWLEDGE FOR A SOFTWARE ARCHITECTURE ORGANIZATION

The DSK model can be applied to describing and predicting the competence of teams of architects and architecture-producing organizations as well as to individual architects. For example, adequately funding the architecture effort is an organizational duty, as is effectively using the available architecture workforce (by propitious teaming, etc.). These are organizational duties because they are outside the control of individual architects. An organization-level skill might be effective knowledge management or human resource management as applied to architects. An example of organizational knowledge is the composition of an architecture-based life-cycle model that all software projects use.[7]

An in-depth survey for **organizational** duties, skills, and knowledge is not feasible because there isn't the same wealth of information resources for architecture organizations as for individuals. Nevertheless, it is possible to list a number of likely candidates in each category, using

- our own experience
- preliminary results from a questionnaire given to practicing architects
- the organizations' architecture improvement efforts to which we are privy
- enterprise architecture competence frameworks and maturity models

From these sources we have drawn up a candidate list of architectural duties for an organization:

- Hire talented architects.
- Establish a career track for architects.
- Make the position of architect highly regarded through visibility, reward, and prestige.
- Establish a clear statement of responsibilities and authority for architects.
- Establish a mentoring program for architects.
- Establish an architecture training and education program.
- Track how architects spend their time.

---

[7] Joe Batman provides a nice example of a prescription of organizational duties in his essay "Hunting the Product Line Architecture." He lists the following organizational characteristics as essential for producing good architectures on a routine basis [Batman 2008]:

- architectures created within a defined-process atmosphere that imposes standards and follows written processes
- the incorporation of architecture design and evaluation into development processes and plans
- management of the development process that reflects the central role of architecture specification
- corporate-level policies, processes, and investments that support the creation and maintenance of sets of common reusable assets
- specific and comprehensive requirements elicitation for architecture
- adequate provision of a process for sustaining the architecture
- quality assurance organizations that are integrated into the process of architecture design and evaluation
- architects or an architect assigned to each project
- a training program for developers and other stakeholders to educate them on the role of architecture
- addressing the cultural barriers associated with architecture-centric development

- Establish an architect certification program.
- Have architects receive external architect certifications.
- Measure architects’ performance.
- Establish a forum for architects to communicate and share information and experience.
- Establish a repository of reusable architectures and architecture-based artifacts.
- Develop reusable reference architectures.
- Establish organization-wide architecture practices.
- Establish an architecture review board.
- Measure the quality of architectures produced.
- Provide a centralized resource to analyze and help with architecture tools.
- Hold an organization-wide architecture conference.
- Initiate software process improvement or software quality improvement practices.
- Have architects join professional organizations.
- Bring in outside expert consultants on architecture.
- Include architecture milestones in project plans.
- Have architects provide input into product definition.
- Have architects advise on the development team structure.
- Give architects influence throughout the entire project life cycle.
- Reward/penalize architects based on project success.

Once we have a degree of confidence in a set of effective duties, skills, and knowledge areas for organizations, an assessment instrument and strategies for improvement should follow as they did for individuals (see Section 7 for more about assessment instruments).

3 The Human Performance Technology Model

If I want to know if people are competent, I have to observe how they behave, don't I? My answer to such questions is a firm "No!"

—Thomas F. Gilbert (1928-1995)

The second model of competence we are exploring comes to us from the world of human performance technology (HPT). A leading contributor to that field, sometimes called its "father," was Thomas F. Gilbert, an engineer who applied his understanding of the process of technological improvement to human beings [Wikipedia 2007a].[8]

Gilbert's best-known contribution to the field was the foundational book *Human Competence: Engineering Worthy Performance,* originally published in 1978 [Gilbert 1978]. A "tribute edition" was released in 1996, one year after Gilbert's death [Gilbert 1996]. He established the basic principles of performance improvement that are used by HPT consultants today (for an example, see www.sixboxes.com/The_Model.html) [Six Boxes 2006]. Gilbert identified six variables that he believed were necessary to improve human performance:

- information, resources, and incentives, which he cataloged as environmental factors for which management is responsible
- knowledge, capacity, and motives, which he cataloged as factors inherent in the "behavior repertory" of the individual

Gilbert believed that it was the absence of performance support at work, not an individual's lack of knowledge or skill, that was the greatest barrier to exemplary performance. Therefore, he believed it was most necessary to focus on variables in the work environment before addressing the individual.

In his article "McGregor meets Gilbert," Nickols points out that drawing a distinction between individual and environment has a long history [Nickols 2007]:

Earlier, Douglas McGregor drew essentially the same distinction when he wrote: "…the performance P of an individual at work in an industrial organization is a function of certain characteristics of the individual I, including his knowledge, skills, motivation, attitudes and certain aspects of the environmental situation E, including the nature of his job, the rewards associated with his performance, and the leadership provided him" [McGregor 1967 p.5].

The same dichotomy between the environment and the individual can be seen in Teodorescu and Binder's model shown in Figure 1 on page 4 [Teodorescu 2004].

[8] The International Society for Performance Improvement's most prestigious award is the Thomas F. Gilbert Distinguished Professional Achievement Award (http://www.ispi.org/awards/2006/honorary2006.htm).

The difference between focusing on behavior and focusing on performance is more than philosophical; it's practical. Gilbert points out that in many cases, small differences in behavior can lead to large differences in performance. Two marksmen may load and hold their rifles, breathe, and sight the target in exactly the same way. But the one who squeezes the trigger will hit the target, whereas the one who pulls the trigger will not. In spite of almost identical behaviors, one is obviously more competent than the other.

The Human Performance Technology model's fundamental aspects are as follows [Gilbert 1996]:

- We wish to engineer *worthy* performance; that is, performance that produces value at reasonable cost. Worth = Value / Cost.
- The Potential for Improving Performance (PIP) is the ratio of exemplary performance to typical performance, where exemplary performance means the best performance for which we could reasonably hope. PIP = $W_{exemplary} / W_{typical}$. If the best worker in the best environment turns in a performance worth 75 units at a cost of 10 units, then that worker's performance is 7.5. If the average worker turns in a performance of 2.5, then we have a PIP of 3, which signals the opportunity to triple what the average performer produces through improvement steps.

  High PIPs are opportunities for great improvement and should be the focus of effort. Low PIPs (near 1) will require a lot of investment to further improve. PIPs in extremely competitive fields tend to be low,[9] as do PIPs for highly repetitive tasks.

  This reasoning assumes (which Gilbert explicitly does) that "poor performers usually have great potential." Or, "the more incompetent a person or a group of people are, the easier it is to improve their performance." The ratio, to be meaningful, must be stated for an identifiable accomplishment, because there is no "general quality of competence."
- The key to performance improvement is finding the right measures that accurately reflect the value of the performance.

Measuring performance of various activities throughout an organization can help you find where the biggest improvements can be made (see Figure 5). However, acting on the PIP may or may not result in economic gain to the organization. It depends on the contribution that activity has to the bottom line. That is, when applying the Human Performance Technology model to improve architecture competence, we should focus on the aspects of the organization that have the most impact on producing high-quality architectures.

Performance has many dimensions; its worth is not uni-dimensional. Gilbert identifies three kinds of measurements, with subclasses [Gilbert 1996]:

1. quality
    a. accuracy: degree to which accomplishment matches a model, without errors of omission or commission

9 Professional sports provide a good example. Gilbert writes that Babe Ruth scored 177 runs in 1921, whereas the average that year was 87.5. By this measure, the PIP = 2.02 for that season, which is one of the highest PIPs observed in sports.

    b. class: comparative superiority of an accomplishment beyond mere accuracy. Possible measures include market value (high-class furniture likely to sell for more), judgment points (as for show dogs), physical measures (such as number of manufacturing flaws), opinion ratings (Oscars, "MVP")
    c. novelty: state or quality of being novel, new, or unique. An engine that gets 100 miles per gallon is novel. For artistic novelty, we probably resort to judgmental points or opinion ratings.
2. quantity (or productivity)
    a. rate: applies when bulk is important and production is time sensitive (pieces produced per hour; time to completion)
    b. timeliness: applies when production is time sensitive, but bulk not important (Cinderella home by midnight, letter mailed by sundown).
    c. volume: applies when bulk is important, but not time sensitive. ("How many fish did you catch?")
3. cost
    a. labor: behavior repositories (direct overhead, benefits, wages, insurance, taxes)
    b. material: environmental support (supplies, tools, space, energy)
    c. management: supervision (management supports, public taxes, internal allocations of administrative costs)

3.1 USING THE HUMAN PERFORMANCE TECHNOLOGY MODEL TO MEASURE AND IMPROVE ARCHITECTURE COMPETENCE

Applying Gilbert's theory to architecture competence involves solving two problems:

1. How do we measure the worth of an architect's performance? In particular this involves measuring the worth of the various artifacts produced both prior and subsequent to the architecture during the life cycle.
2. What sort of organizational and measurement infrastructure is necessary to calculate the worth? Even once we know **what** to measure for worth we still must determine **how** to measure it. What are the possibilities in terms of routine measurement, how are these results maintained, and how are they used to improve an organization's architecture competence?

As yet, we have only preliminary thoughts on the first question and none to report on the second. We propose to use the duties from the DSK model to isolate the various aspects of the architect's job. If we are auditing an organization's architecture accomplishments, we can identify the job

Figure 4: Architecture Job Accomplishments

accomplishments using the duties from the first column of Table 1 on page 16, shown in Figure 4. We can refine each of these duties into task accomplishments using the second column of Table 1. There is no claim that each of these carries the same value, but each carries **some** value that can be used to measure current performance against an exemplar.

Gilbert also introduces a Behavior Engineering Model, showing the things we can do to increase competence through greater behavior efficiency. The model is shown in Figure 5.

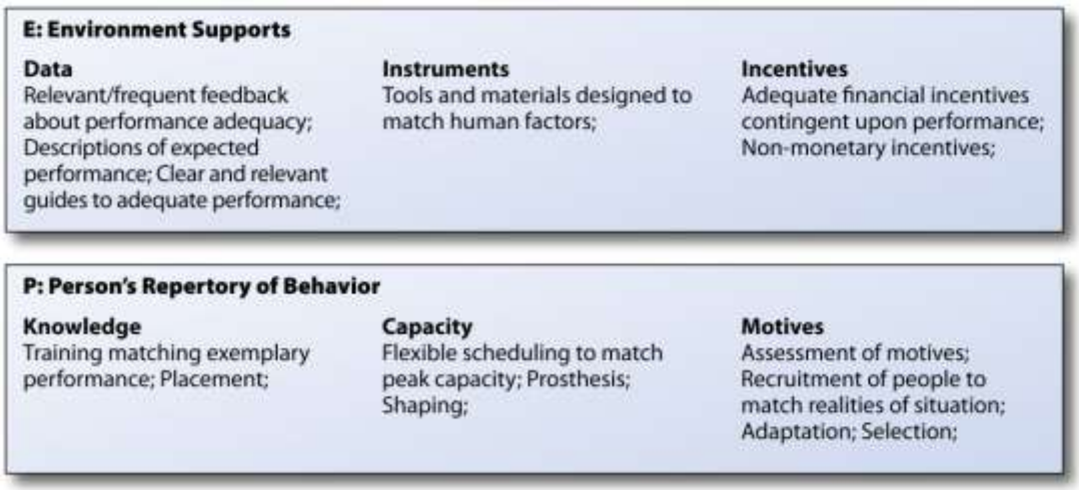

Figure 5: Gilbert's Behavior Engineering Model [Gilbert 1996]

Gilbert prescribes the following, in the order given, to improve competence:

1. Begin with "Data" (row 1, col 1). Do the architects know how they should perform and how they have performed?
2. Examine the tools and materials (row 1, col 2) the architects work with. If these can be improved, much training can be eliminated.
3. Look at incentives (row 1, col 3). How can they be improved and made contingent on good performance?
4. Finally—though not least important—look at training (row 2, col 1). Training is very powerful, but often very expensive.

This order is the most efficient one, because it is likely to get the most improvement for the least cost. Further, improving one area is likely to benefit untouched areas. For example, improved incentives may lead people to learn more even when there's been no effort to teach them better.

4 The Organizational Coordination Model

Teams at multiple sites developing a single product or a related set of products must cooperate to produce a functioning product. The cooperation's external manifestation is the coordination of activities. The goal of this model is to help us understand, within a particular organizational setting

- what coordination activities are brought on by particular architectural decisions
- how effective are particular organizational coordination mechanisms

Understanding the necessary coordination that results from classes of architectural decisions will yield the coordination depth and volume associated with an architecture. Understanding the effectiveness of particular coordination mechanisms will yield the coordination potential of an organizational structure.

Figure 6 shows two modules of an architecture that share a dependency. It shows the development team for each module and represents the teams' coordination. *Coordination* is defined as "the harmonious combination or interaction, as of functions or parts" [Random House 2006]. The harmonization can be done explicitly (through coordination meetings or various types of communication), or it can be done implicitly (through documents, processes, or common understanding of ongoing tasks).

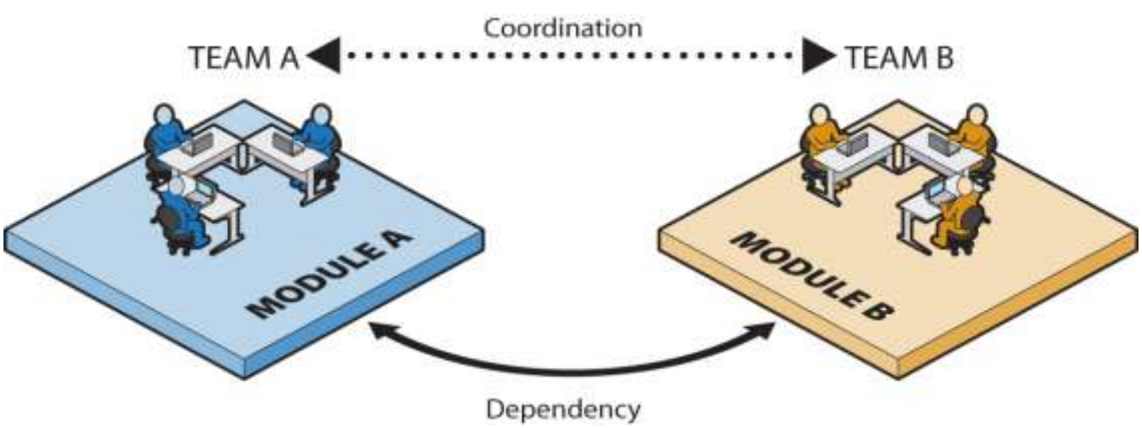

Figure 6: Dependencies Between Modules with Coordination Between Development Teams

4.1 DEPENDENCY

The normal method for dividing large segments of a system into units is to collect them into modules that have minimal interaction. A Dependency Structure Matrix (DSM) is a common technique for representing the interaction of the various modules.

Wikipedia describes DSM as follows:

> *A **Dependency Structure Matrix**, or **DSM** (also referred to as Dependency Structure Method, Design Structure Matrix, Problem Solving Matrix [PSM], incidence matrix, N-square matrix or Design Precedence Matrix), is a compact, matrix representation of a system or project. The approach can be used to model complex systems in systems engineering or systems analysis, and in project planning and project management.*
>
> *A dependency structure matrix lists all constituent subsystems/activities and the corresponding information exchange and dependency patterns. In other words, it details what pieces of information are needed to start a particular activity, and shows where the information generated by that activity leads. In this way, one can quickly recognize which other tasks are reliant upon information outputs generated by each activity.*
>
> *DSM analysis provides insights into how to manage complex systems or projects, highlighting information flows, task sequences and iteration. It can help teams to streamline their processes based on the optimal flow of information between different interdependent activities [Wikipedia 2007b].*

As indicated by this description, determining whether a dependency exists and a dependency's type is not necessarily a simple exercise. Dependencies can be activity based or information based. These dependencies are determined by examining the runtime behavior of the system. Teams, on the other hand, develop modules (a static portion of the system). The determination of a dependency, then, involves understanding the runtime behavior of a system and working back from that to the modules.

Our study of the organizational coordination component of architecture competence will focus on the techniques used by an organization to determine the dependencies among modules, understanding that the architecture evolves during the development process.

4.2 THE COORDINATION CAPABILITY OF AN ORGANIZATION

Figure 7 shows some of the factors that determine an organization's coordination capability. An organization coordinates through its structure, its processes, and the tools used to support the coordination. When the different development teams are colocated, opportunities exist for various informal coordination mechanisms, such as meeting in the lunch room or at social events, that are not available when the teams are in different geographical locations.

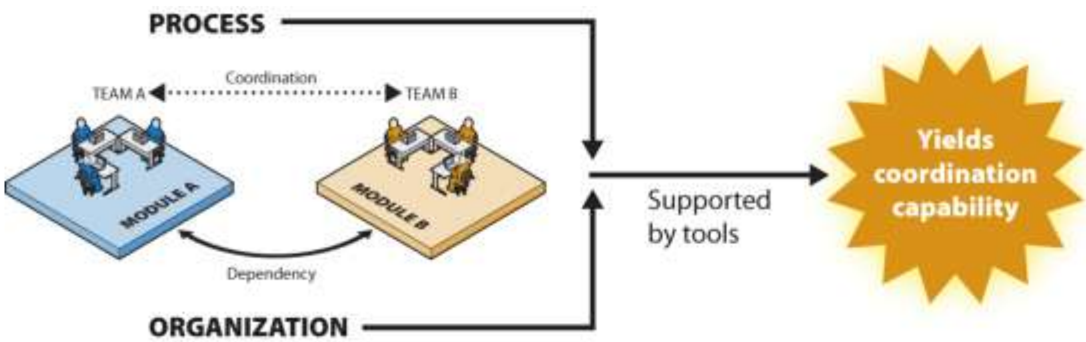

Figure 7: Factors that Affect Coordination Capability of an Organization

An organization's ability to match the coordination capability with the coordination requirements imposed by the architecture is a facet of architecture competence. Coordination among the development teams allows decisions to be made with an appreciation of the context and the implications. Suppose Team A and Team B must both make decisions about how to manage the granularity of an image in their modules. If their decisions are consistent, image integration will be smoother than if their decisions are inconsistent. Options for supporting consistent decision making between the relevant developers include using an intermediary, having a facilitator, or enabling the relevant developers to communicate directly. Each of these options introduces overhead and provides benefits in terms of sharing knowledge, bringing additional contextual information to bear on the decision, and utilizing the decision to make future decisions. For example, some benefits of having an architect as an intermediary between the two sets of developers are the availability of additional contextual information and the ability to utilize the decision when making future decisions. Some costs of an intermediary include the possibility of bottlenecks, the potential of introducing delays, and the potential loss of information when it is passed between developers.

This is one example of how an organization might influence the coordination capability. There are similar examples in terms of the development processes that are engaged and the tools that are used to support the coordination aspects of the development activity.

4.3 MEASURING THE COORDINATION ACTIVITIES

Since the relation between coordination activities and architecture impacts architecture competence, it is important to understand how to measure the coordination. We have already discussed the use of DSMs as a means for measuring the requirements for coordination. Measuring the coordination activities among teams can be done across several axes, for example

- the coordination activity between teams and various artifacts. For example, how frequently is the architecture documentation accessed by various teams? This data is usually maintained by the content management system.
- the coordination activity related to particular known problems. This activity can be measured by examining discussion board posts. How many posts does it take to solve problems, and what is the duration of the discussions?

Surveys of developers can also be performed at a small cost. Conducting a five-minute survey every month or so can make available data about whether problems are quickly resolved and what developers consider the main impediment to getting the information they need to solve their problems.

4.4 RELATING ORGANIZATIONAL CAPABILITY TO DEPENDENCIES

The two aspects we have discussed—(1) dependencies within an architecture and how they evolve and (2) organizational coordination capability—should be aligned within an architecturally competent organization.

We would expect, when looking at an organization that is architecturally competent, to see evidence that the following types of concerns have been addressed within a development project:

- Dependencies and their evolutionary paths are identified.
- The choice of coordination mechanisms used among various teams is appropriate to the type of dependencies among the modules being developed by those teams.
- The choice of coordination mechanisms used among teams during development changes is appropriate to the portions of the modules being developed during particular increments.

Our point is not that one organization structure or set of processes is better than another for performing multisite development but that the interaction of the architecture, the organization structure, the development processes, and the tools used for coordination affect the time required to develop the system from the architecture (Gilbert's Quantity Measure) as well as the accuracy of the architecture (Gilbert's Quality Measure).

5 The Organizational Learning Model

The final model that we will consider comes from the field of organizational learning. Argote and Todorova summarize and provide a framework for the research performed in this field since 1995 [Argote 2007]. *Organizational learning* is defined as a “change in an organization that occurs as a function of experience.” This change could be reflected in various ways including in changes to the organization’s knowledge, its routines, or its performance.

Learning processes transform experience into knowledge, moderated by context. The framework has four classes or variables shown in the lower half of Figure 8: experience flows, learning processes, knowledge stock, and organizational context [Argote 2007]. The figure as a whole shows the relationship between organizational learning and some important architecture-centric practices. Carrying out the duties associated with the transformations between artifacts and the verifications of those transformations provide the architecture-centric experiences. Learning processes will transform these experiences into knowledge stock. Given that an organization has suitable learning processes, it will be able to exploit this knowledge stock in future experiences. This cycle shows how organizational learning contributes to an organization’s architecture competence.

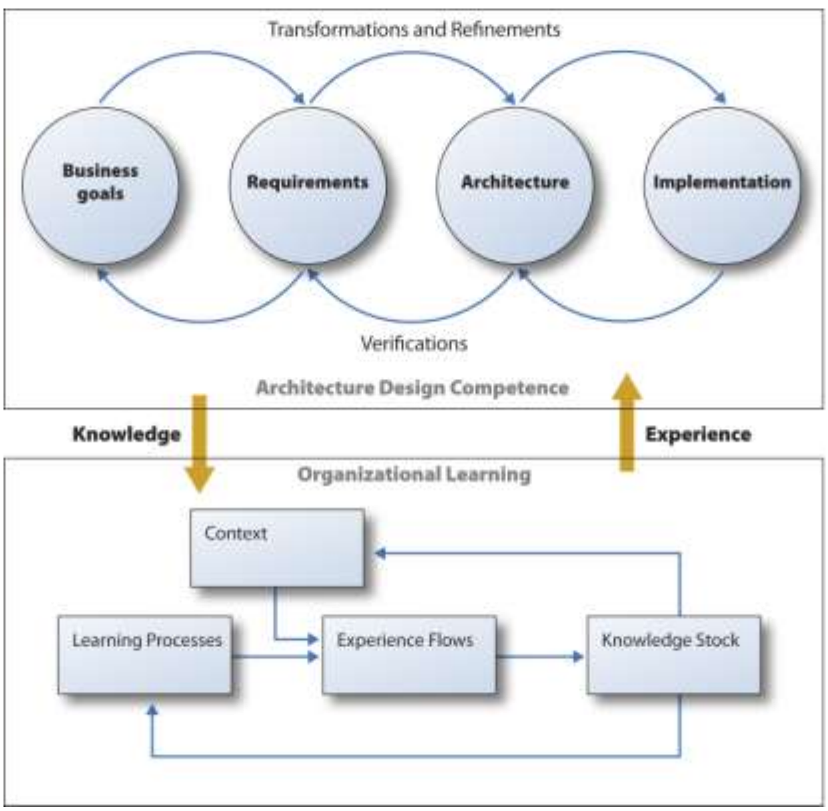

Figure 8: The Relationship Between Architecture Design Competence and Organizational Learning

5.1 THE COMPONENTS OF THE ORGANIZATIONAL LEARNING FRAMEWORK

According to the Argote/Todorova framework, the antecedents, processes, and outcomes of learning consist of experience, the context, learning processes, and the knowledge stock [Argote 2007]. And these antecedents, processes, and outcomes work at three levels: individual, group, and organizational. The Argote/Todorova definition of *context* includes the higher level for each of the levels of learning that the framework describes. For example, for individual learning, the context includes the groups in which the individual is embedded. The influence of the lower level on the learning processes occurs through experience. For example, both group experiences and individual experiences are inputs to group learning processes.

Experience is what happens to the individual, group, or organization in performing the task, while *knowledge* is the outcome of the processing of the experience via learning. Experience might be measured by the cumulative number of tasks performed. The *flow* of experience in an organization can be characterized along several dimensions that can operate at the individual, group, or organizational levels. One important dimension of experience is whether it is direct—based on an organizational unit's own direct experience, or indirect—based on the experience of others. Another dimension is content: is the experience about tasks or about relationships? Experience can also be characterized spatially, as discussed in Section 4 of this report: it can be acquired in close spatial proximity (colocated teams, for example) or it can be acquired across distant locations. Other dimensions of experience include novelty, heterogeneity, and ambiguity. Experience can also be characterized in terms of whether it is a success or failure. Finally, the flow of experience can be characterized along temporal dimensions, including its pace and how recent it is.

An architecturally competent organization will understand the organizational learning opportunities presented by the various types of experiences in performing architecture-centric practices [Argote 2007].

Organizational **learning processes** transform experience into **knowledge**. The "mindfulness" involved in the experience affects how it is processed. Mindful processes involve attention, whereas less mindful processes are driven by routines or rules. Mindful processing is typically under the individual's control; less mindful processing occurs without an individual's active control. For example, an architecture team that conducts a review or architecture analysis after completing a module or subsystem to learn what went well and what went wrong exemplifies a group engaging in mindful learning. Learning processes also vary according to whether the learning occurs directly as a result of the unit's own experience or indirectly through the experience of others. Organizations may learn from their own direct experience, or they may learn from the experience of other units. The architecture team reviewing its own experience is learning from its own experience. A team that consults another team and adopts its superior practice is learning from the experience of others. This form of organizational learning is referred to as *knowledge transfer*.

An architecturally competent organization will strive to understand which types of learning processes are best suited for different types of experiences [Argote 2007].

The *learning context* consists of two main components: the organization and the environment. Both formal and informal aspects of the organization affect organizational learning. Formal organizational arrangements, such as technology and structure, influence organizational learning and knowledge transfer, as discussed earlier in this report. But informal aspects of the organizational context, such as culture and social networks, also influence learning processes. Learning processes occur in communities of practice that may cut across organizational boundaries. The volatility and heterogeneity of the environment also affect learning processes and outcomes. In highly dynamic environments where the learning situation is changing dramatically all the time, the causality of events may be difficult to ascertain. The constraints on learning can lead to a bias towards less mindful learning processes and to "superstitious" learning characterized by inappropriate inferences being drawn from experience. After knowledge is created through learning processes, it is embedded in a reservoir or repository in the organization. For example, knowledge acquired by the architecture team could be embedded in a new method, a tool, or an individual team member's understanding of some aspect of the procedure.

An architecturally competent organization will strive to understand how various types of learning contexts affect the transformation of experience into knowledge stock [Argote 2007].

5.2 USING THE ORGANIZATIONAL LEARNING FRAMEWORK TO MEASURE AND IMPROVE ARCHITECTURE COMPETENCE

Measuring organizational knowledge is a challenge. Historically, three approaches have been taken to measuring it:

1. Measure directly though questionnaires, interviews, and verbal protocols, scoring these to quantify differences between individuals or to note learning over time. An exam that tested an architect's knowledge of architectural styles, analysis techniques, and quality attributes might be an example of such a measurement instrument.
2. Regard changes in routines and practices as indicators of changes in knowledge. If a group changes how it reviews an architecture prior to each major development stage, that might indicate that organizational learning has taken place.
3. View changes in organizational performance indicators associated with experience as reflecting changes in knowledge. For example, if a team becomes better at delivering its products within budget and within schedule, we would infer that the team has learned better methods than it engaged in the past.

6 Considering the Models Together

6.1 HOW THE MODELS TOGETHER SUPPORT EVALUATION

In Section 1, we asserted that the “end game” of competence was to carry out architecture-centric practices reliably and efficiently, and to improve that capability across systems and over time. How do the models help us evaluate an individual’s or organization’s ability to do that? The practices are carried out by various groups of people using processes and tools. The models will let us examine these from different perspectives from which will emerge an overall picture of competence.

Clearly duties, skills, and knowledge (on the part of organizations as well as individuals) are involved in the practices we described in Section 1.2; those practices can be cast as duties, while supporting skills and knowledge are necessary for successful completion of the duties.

To effectively perform practices and duties, individuals and groups must have a repository of accumulated knowledge and experience. The Organizational Learning model provides a way to evaluate how effective that repository is. It also tells us how “mindful” the learning needs to be.

The organizational coordination model, in practice, concentrates most heavily on the practice that produces an architecture-conformant implementation, and tells us how effective the organization is likely to be at carrying out that practice. Since fielding a system is the ultimate point of building an architecture, it is reasonable to have an architecture competence model that focuses on that activity.

Behind all of this lies the HPT model that tells us how to value the artifacts produced by the practices. A “Gilbertian” view of the world will let us apply value to the answers we get from our questions, and gives us the hypotheses we need to evaluate.

Given the four models, it is tempting to try to produce a “grand unified theory of competence” by somehow combining them. For example, it is easy to see how the DSK and HPT models can be usefully combined. HPT calls for calculating the value or worth of specific job accomplishments so that low-performing areas can be targeted for improvement. Improvement, in turn, depends on identifying the steps involved in carrying out a task. Both of these are handily served by the DSK model, which supplies the list of job accomplishments and task steps in the form of duties. One specific HPT improvement strategy is to provide performers with information that they internalize as knowledge. What kind of knowledge? Again, the DSK model tells us. In this combination of the HPT and DSK models, HPT is dominant and takes input from DSK when needed.

Similarly, the Organizational Learning model is concerned with how organizations internalize and utilize knowledge. One way organizations do this is by building and nurturing appropriate and

effective coordination mechanisms. Testing how well they've done this to achieve an end (in this case, producing high-quality architectures[10]) can be accomplished through HPT methods. In this combination of models, organizational learning is dominant and "calls" the other three to inform it in specific areas.

For now, we have chosen not to produce a "unified model of competence," focusing instead on using the strengths of each of the models to inform our competence evaluation instrument and improvement strategies.

6.2 PRINCIPLES EMBODIED BY THE MODELS

If the end game of competence is to predictably and repeatably produce high-quality architectures by effectively carrying out the important architecture-centric practices, it is imperative that each model be clear on how it contributes to that goal. Viewed in this light, each model can be seen to embody a set of assumptions or hypotheses—we'll call them principles—where a principle takes the form

> *X is likely to lead to high-quality architectures because Y.*

For example, the DSK model implicitly assumes that carrying out the listed duties, possessing the listed skills, and having the listed knowledge is more likely to lead to high-quality architectures. For instance, a prescribed skill from the DSK model is the ability to handle the unknown, which reflects the following principle:

> *Possessing skills to handle the unknown is likely to lead to high-quality architectures because the architect will be able to effectively identify areas of uncertainty and be equipped to take the appropriate steps to either eliminate the uncertainty or make the architecture flexible to accommodate it.*

A duty, such as documenting the architecture, immediately suggests this principle:

> *Documenting the architecture is likely to lead to high-quality architectures because documentation is essential to effective communication, which is essential to effective understanding and use by the architecture's stakeholders, which is in turn essential to providing timely and useful feedback.*

A seemingly "extracurricular" duty such as mentoring other architects suggests this principle:

---

[10] A *high-quality architecture* is one that predictably enables the creation of a system that satisfies the producing organization's business goals.

Mentoring other architects is likely to lead to high-quality architectures because being mentored is an effective way to gain real-world experience and thus become a more capable architect.

The principles behind HPT start with the following overarching one:

Measuring the worth or quality of architectures produced to date is likely to lead to high-quality architectures because instances of poor performance will be readily identified, which will lead to the examination of the causes, and eventually to remediation and improvement.

The principles relating effective organizational coordination and organizational learning to high-quality architectures are also apparent. These principles amount to prescriptions. Highly competent individuals, teams, and organizations will observably exhibit aspects that (the models posit) will lead to high-quality architectures.

6.3 COVERAGE PROVIDED BY THE MODELS

How well do the four models serve us when we consider their use in an evaluation instrument? One way to judge this is by plotting the coverage they bring to the problem. At least three kinds of coverage are important. The first is coverage of the kinds of artifacts that can be observed and evaluated. The second is whether the model applies well to individuals, teams, or whole organizations. The third is whether the models tell us to examine past performance or present activity.

Coverage of observables is described below:

1. There are artifacts that the subject has produced. These can be architectures as represented by models or documents. They can also be systems that can be seen to run, or that can be represented by source code listings.
2. There are processes that the subject carries out. Processes can be observed to see what the steps are, how well each step is executed, and whether tools and technology are effectively employed.
3. There is the organization itself. The organization's structures, interactions, and coordination can be observed.

Each of the four models will lead us to observe one, two, or all three of these things with more or less emphasis. As Table 5 illustrates,[11] together the four models cover the three categories well, with each category having at least one model that pays it the most attention.

[11] The table covers artifacts—meaning architectures—processes, and organizations. This description of coverage types closely tracks the Philips Research BAPO model, where *B* stands for business concerns, and *A*, *P*, and *O* are as formulated here [America 2003]. Where do our models stand with respect to business concerns? That is, should Table 5 have an extra column showing business as an element of coverage by the models? We do address business concerns; producing systems aligned with business goals is how we define a high-quality architecture. Our evaluation instrument will assess and improve A, P, and O to predict and control architecture competence. We will only be concerned with how A, P, and O interact with B; we do not strive to assess or improve B as a means to predict and control competence.

Table 5: Priority Each Model Assigns to Observing Artifacts, Process, and Organization

Model	Artifacts	Process (including enabling tools and technology)	Organization
DSK model	Tertiary. For example, architecture documents may be observed to infer how well the duty of documentation was carried out.	Primary. Examines duties of architects and architecting organizations.	Secondary. Some organizational duties involve coordination and institutionalization of information, such as repositories of past designs and artifacts.
HPT model	Primary. Architectures and systems are examined for their "worth" and alignment to business goals.	Secondary. Steps in the process can correlate to task-level accomplishments and areas to target for improvement strategies.	Tertiary. Evaluating worth or value of output depends on measuring against the goals of the appropriate part of the organization.
Organizational Learning model	Secondary. Some learning artifacts amount to repositories of learned knowledge.	Secondary. Some learning results from the way that processes are carried out.	Primary. Emphasis is on how organizations take in, retain, and utilize knowledge.
Organizational Coordination model	Secondary. Some coordination (or lack of it) manifests itself in the quality of the artifacts.	Secondary. Some coordination mechanisms may be embodied in processes.	Primary. Emphasis is on what coordination activities are caused by particular architectural decisions

Coverage of individuals, teams, and organizations: Table 6 rates the models in terms of how well each one applies to individual competence, the competence of teams, and the competence of organizations. A "+++" means the model is very applicable, and a "+" means it is somewhat applicable.

Table 6: Applicability of the Models to Individuals, Teams, and Organizations

Model	Applicability to individuals	Applicability to teams	Applicability to organizations
Duties, Skills, Knowledge	+++	++	+
Human Performance Technology	+++	+++	+++
Organizational Coordination	+	++	+++
Organizational Learning	++	++	+++

Coverage of past performance and present activity: The HPT model clearly tells us to evaluate past performance. The DSK model clearly tells us to evaluate present activity. The other two models suggest some of each. Both learning and coordination can be evaluated by examining the observable model processes as well as the results of carrying them out.

We found that together the four models provide strong coverage across these important dimensions, giving us confidence that they are effective choices for informing an evaluation instrument. We turn to that in the next section.

7 Building an Assessment Instrument

In this chapter we describe and exemplify the process of building an instrument for assessing architecture competence based upon the four models that were described in Chapters 2-5. We will not present a fully fleshed-out instrument here, for three reasons: 1) Examining such an instrument would involve long and relatively tedious reading; 2) The instrument is not simply a checklist where one runs through every question. It is created as a structured interview where a positive answer might simply cause the interviewer to proceed to the next category of questions, and a negative answer might cause the interviewer to probe more deeply to recursively unveil new sets of questions; and 3) The instrument is not yet complete.

We will present the rationale for creating an assessment instrument for architecture competence, a set of questions that we developed during our consideration of the four models, as well as examples of the relationships between questions. We will also show how one question/answer pair might lead to others.

7.1 ASSESSMENT OUTCOMES

What are some potential outcomes from an assessment? We envision at least three sets of outcomes from assessments of architecture competence, based on who is doing the assessment, who is being assessed, and the goals of the assessment:

- There are outcomes relevant to an **acquisition organization**: such an organization can use an assessment of architecture competence to assess a contractor in much the same way that it might scrutinize a contractor with respect to its SEI Capability Maturity Model® Integration (CMMI®) level [Garcia 2007], or to make a choice among competing bids. All other things being equal, an acquiring organization would prefer a contractor with a higher level of architecture competence, since this typically means fewer future problems and less reworking [Boehm 2007]. An acquisition organization might assess the contractors directly, or hire a third party to do the assessment.
- There are outcomes relevant to **service organizations**: such organizations might be motivated to maintain, measure, and advertise their architecture competence as a means of attracting and retaining customers. They might typically rely on outside organizations to assess their level of competence in an objective fashion.
- Finally there are outcomes that are relevant to **development organizations**: these organizations would be doubly motivated to assess, monitor, and, over time, increase their levels of architecture competence. This would 1) aid in advertising the quality of their products, and 2) aid in their internal productivity and predictability. In these ways, their motivations align with those of service organizations. Businesses regularly assess their own performance in var-

® Capability Maturity Model and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

ious areas—technical, fiscal, and operational (for example, consider the widespread use of multi-criteria techniques such as the Balanced Scorecard or Business/IT alignment in industry [Kaplan 1996, Luftman 1993]). Reasons for these assessments. include determining whether they are meeting industry norms and gauging their progress over time in meeting business/organizational goals.

7.2 THE FOUNDATIONS AND STRUCTURE OF THE INSTRUMENT

To create the assessment instrument, we have taken both top-down and bottom-up approaches. Top-down, we have generated questions from a knowledge of the place of architecture in the software development and system development life cycles. For example, we know that architectures are critically influenced by quality attribute requirements, so questions in the instrument must probe the extent to which the architect[12] elicits, captures, and analyzes such requirements. Bottom-up, we have worked from the specific duties, skills, and knowledge in the DSK model, and from the components of the organizational learning, organizational coordination, and human performance models. In the bottom-up approach, we examine each category in the models and generate questions that address each component. This approach leads to tremendous overlap, which helps to validate and refine the questions in the instrument.

To take the example introduced above—that of capturing and analyzing quality attribute requirements—we have listed below several duties from the DSK model that directly speak to this issue:

- Capture customer, organizational, and business requirements for the architecture.
- Articulate, refine, and document architectural requirements, ensuring that they meet the company's needs.
- Work with designers, technologists, and researchers to ensure that the user interface reflects client, user, and design requirements.
- Advise the project manager on the tradeoffs between software design choices and requirements choices.
- Reevaluate the architecture for implementing use cases and other requirements such as performance and scalability.

In addition to this list, we consider questions from the other models. From Gilbert's Human Performance Technology model we consider quality attribute requirements from three dimensions: quality (accuracy, class, and novelty), quantity (rate, timeliness, and volume), and cost. This generates questions about the accuracy of the volume and of quality attribute requirements captured, and the cost of doing so. For example, we might ask **how many** quality attribute requirements the architect collects, how the architect knows that this number is enough, how quickly the information is collected, and/or how much the requirement elicitation/validation costs. We might also ask questions about whether the collected quality attribute requirements were **novel**—did they add value and insight to the project? And we might ask about the **class** of the collected quality attribute requirements—Is it a factor that contributes to the value of the end product?

12 Whenever we speak of *the architect* in this report, we are referring to an *architecture team*. We see both individual architects and teams, depending on the cooperating organization and the size of the project.

Having generated and validated a large number of questions from the four models, we must then structure the questions, since the assessment instrument is delivered as a series of interviews. We are building a flowchart-style assessment instrument: Each question may lead to other questions that further probe a particular area.

7.3 SAMPLE QUESTIONS

As described in the previous section, we have generated questions from many sources. Some are related to duties, some are related to architecture groups, some are related to a consideration of the different roles in an organization (architect, manager, programmer, etc.), and so forth.

Whenever we pose a question in the assessment instrument, there are a number of **meta-questions** that automatically accompany it; for example

- What is your supporting evidence?
- How sustainable is your answer over time, across different systems, and across different architects? (Sustainability can be prosecuted by asking, for example, the following: How repeatable is this with a different architect? What knowledge is embedded in your processes, tools, and so forth, that needs to be consistently manifested by different architects? How would you nurture your next super-architect?)

With these meta-questions in mind, we now turn to a sample set of questions drawn from the four models. For each model we will describe the source of the question, the question itself, and sub-questions/probing questions that follow from it.

Questions Based on Duties

Duty: Creating an architecture
Question: How do you create an architecture?

- How do you ensure that the architecture is aligned with the business goals?
- What is the input into the architecture-creation process? What inputs are provided to the architect(s)?
- How does the architect validate the information provided? What does the architect do in case the input is insufficient/inadequate?

Duty: Architecture evaluation and analysis
Question: How do you evaluate and analyze an architecture?

- Are evaluations part of the normal software development life cycle, or are they done when problems are encountered?
- Is the evaluation incremental or "big bang?" How is the timing determined?
- Does the evaluation include an explicit activity relating architecture to business goals?

- What are the inputs to the evaluation? How are they validated?
- What are the outputs from an evaluation? How are the outputs of the evaluation utilized? Are the outputs differentiated according to impact or importance? How are the outputs validated? Who is informed of what outputs?

Duty: Life-cycle phases: Testing
Question: How does your architecture facilitate testing?

- Do you have specific architectural facilities to aid in fault detection, recording, playback, insertion, and so forth?
- Do you define any test cases based on the architecture?

Questions Based on Knowledge

Knowledge: Architecture concepts
Question: How does your organization ensure that its architects have adequate architectural knowledge?

- How are architects trained in general knowledge of architecture?
- How do architects learn about architectural frameworks, patterns, styles, standards, documentation notations, and architecture description languages?
- How do architects learn about new or emerging architectural technologies (e.g., model-driven architecture)?
- How do architects learn about analysis and evaluation techniques and methods?
- How do architects learn quality-attribute-specific knowledge, such as techniques for analyzing and managing reliability, performance, maintainability, and security?
- How are architects tested to ensure that their knowledge levels are adequate, and remain adequate, for the tasks that they face?

Questions Based on the Organizational Coordination Model

Question: How is the architecture designed with distribution of work to teams in mind?

- How available is the architecture? How broadly is it shared with various teams?
- How do you manage the evolution of architecture during development?
- Is the work assigned to the teams before or after the architecture is defined, and with due consideration of the architectural structure?

Question: What domain knowledge exists on the various teams?

- How is this knowledge captured and shared?

Question: Are the aspects of the architecture that will require significant inter-team coordination supported by the organization’s coordination/communication infrastructure?

- Do you colocate teams with high coordination? Do you at least put them in the same time zone?
- Must all coordination go through the architecture team?

Questions Based on the Human Performance Technology Model

Question: Do you track how much the architecture effort costs, and how it impacts overall project cost and schedule?

Question: Do you track the “worth” of the architecture and the benefits, such as stakeholder satisfaction, quality, repeat business, and bug reports?

Questions Based on the Organizational Learning Model

Question: How do you capture and share experiences, lessons learned, technological decisions, techniques and methods, and knowledge about available tools?

- Do you use any knowledge management tools?
- Is architectural knowledge embedded in your processes?
- Where is the information about “who knows what” captured, and how is this information maintained?
- How complete and up-to-date is your architecture documentation? How widely disseminated is it?

Question: Are architects actively taught, encouraged, and mentored?

- How are new team members brought up to speed on the architecture? How long does this typically take?
- Do you use reviews to learn from successes and failures?
- Are team members free to, and encouraged to, express their views?

7.4 REFLECTIONS ON THE INSTRUMENT QUESTIONS

What we have presented here is just a small selection of questions evoked by the four models that we are using to understand architecture competence. Even in a small sample, we already see several interesting results. First, there is a fairly straightforward mapping from the elements of the models to questions. Given that there is a duty called “documentation,” we expect to see one or more questions related to documentation.

But in addition to this straightforward mapping from the models to the questions, we can see that the intersection of the models and the combination of bottom-up and top-down approaches work to both flesh out and validate the questions. For example, we not only get questions on documentation coming from the duties, but we also get documentation questions coming from the Organi-

zational Learning model. In that model, we are concerned with how information is collected and shared, and this naturally leads to a question concerning documentation.

There is still a considerable amount of structuring required to turn this set of questions into a document that will guide a competence assessment interview. But the foundations for an assessment instrument have been laid, as we have shown, by delving into our four models of competence.

8 Summary

We’ve presented an approach for evaluating and improving the architecture competence of individuals or organizations. The approach is based upon our mining of four models that have shown relevance to the area of architecture competence. For each we have shown how the model can be used to evaluate the competence of an individual or an architecture-producing organization, and how the model might be used as the basis for competence improvement. Finally, we have shown how the models lend themselves straightforwardly to the production of an architecture competence evaluation instrument. The instrument is based on the principles and observational requirements implied by each of the models.

Together the four models provide strong coverage in a number of ways. They provide a basis for measuring past performance as well as current activity. They cover the range of observational possibilities: artifacts, processes, people, and organizations. Finally, they apply well as a group to individuals, teams, and organizations. This coverage gives us confidence that the four models together will produce valuable and reliable results.

8.1 NEXT STEPS

Specific next steps include the following:

1. **Survey practicing architects.** Appendix A shows a survey form that we have circulated (and continue to circulate) among various architect communities. The survey will add to our understanding of important duties, skills, and knowledge, and also tell us ways in which an organization can improve its competence. We also need to determine how the data will be analyzed and how the results of that analysis will be expressed.
2. **Pursue the DSK model in more depth.** After we refine the DSK model, a valuable exercise would be to repeat the community survey and compare the results for consistency. Also, more in-depth analysis could be used to investigate the following questions:
    a. Are there duties that are considered essential?
    b. What are the most important or critical skills?
    c. What is critical knowledge? Which is domain dependent? Which is domain specific?
    d. Do DSK models vary by geographical region?
    e. Are particular DSK models more closely associated with evolving architectures than with building a system from scratch?
    f. Do architects in larger organizations have different DSK models than those in smaller organizations?
    g. Do architects of relatively large systems have different DSK models than those of relatively small systems?

h. How much do DSK models reflect conceptual (timeless) knowledge, as opposed to knowledge of specific tools, methods and techniques currently in vogue?

3. **Investigate the importance of "experience" in an architecture staff.** For example, is an organization with a smattering of highly experienced people in any sense more competent than one without it?

4. **Account for different kinds of software organizations.** An architect at a major IT service organization was once heard to say, "My biggest architectural decision is whether we choose SAP or Oracle." The competence required to make that decision may be quite different from the competence required to architect a real-time embedded military command and control system. To generalize, there are different kinds of organization/architecture relationships. There are end-user organizations that buy software (imbued with architecture) and use it, and they need the competence to recognize good (or bad) architectures and react accordingly during procurement or acquisition. There are supplier organizations that produce products for end-user organizations, and they need architecture competence to produce high-quality architectures for their products. Finally, there are technology platform organizations that build technology platforms or other software "commodity" software products, and they need a related but different kind of architecture competence. The supplier organization in the middle is an "end-user" organization of the product/platform/technology organization. Different architects evaluate those products and then produce their own products, and as a result different competences are needed. Figure 9 depicts this situation.

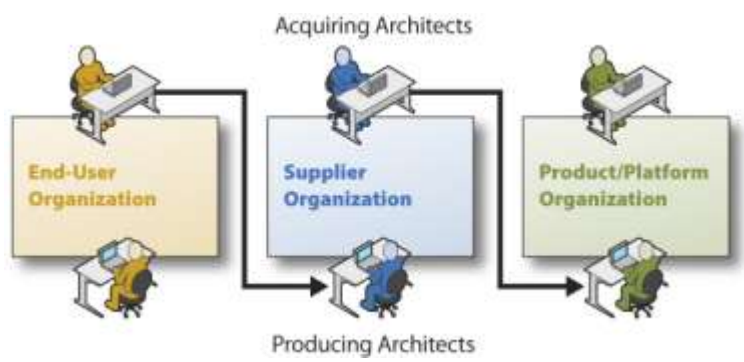

Figure 9: "Acquiring Architects" in One Kind of Organization Interact with "Producing Architects" in Another

5. **Establish the relationship between this work and the CMMI approach.** The CMMI approach sets out broad categories for an engineering process, but does not provide much technical detail about how to carry out each process. In any case, our work has a decidedly architecture-centric focus, which CMMI does not emphasize. We view this work as complementing CMMI. Whereas CMMI concentrates more on providing general process requirements for various topics, we see ourselves as providing specific strategies for architectural measures, capabilities, and improvement strategies. In the long term, we would like to pilot our measurement and improvement approaches in high-maturity organizations to see if any favorable changes can be observed.

6. **Write a case study of a competence-improving organization.** Some organizations have very well-defined competence programs for their architects now. Other companies have introduced competence-enhancing activities even if they do not use that term. A case study that illustrates how competence activities are introduced and matured could help other organizations adopt similar approaches.

8.2 CONCLUSION

As a last word, this work has historical precedence. Pritchard writes compellingly how France became a world-class naval power only after elevating the profession of (what we would call) naval architect to a highly visible position with a rich body of institutionalized knowledge and education [Pritchard 1987]. We believe that, within software engineering, the field of architecture is no less strategically important. Our work is about helping architects do what they do better. We expect this to pay rich dividends.

Appendix A: Survey of Practicing Architects

Purpose: The Carnegie Mellon Software Engineering Institute is conducting a survey of software architects, as part of a project dealing with understanding and improving software architecture competence. For more information, visit www.sei.cmu.edu/architecture/competence.html.

Survey guidelines: Your participation is completely voluntary. There will be no financial or other benefits from participating, other than our thanks and knowing that you are helping further the study of the field.   Your responses will be kept confidential and not be used outside of Carnegie Mellon University in any way that can identify you personally.

Instructions: Type your answers in the indicated fields. Click on the radio buttons to record multiple-choice answers. If you are a consultant, try to answer organization questions using the typical or most important organization for which you provide architecture services.

When you have completed the survey, please return it by email to clements@sei.cmu.edu.

About you

A1. Your name [optional]	
A2. Are you currently an active software architect?	☐ **Yes** ☐ **No**
A3. What is your job title?	
A4. On how many projects have you acted (a) as chief software architect? (b) as non-chief software architect? (c) as architecture consultant to a project?	(a) Chief: (b) Non-chief: (c) Consultant:
A5. How many years experience do you have (a) as a software architect? (b) as a software professional?	(a) Architect: (b) Professional:

A6. What is the size of projects in which you've acted as architect or architectural consultant? Use whatever measurement scale you wish. You may give an average or typical size, a range of sizes, or a list of sizes.	
A7. How long do you typically spend on projects for which you act as architect or architectural consultant? Try to cite milestones (e.g., from contract proposal to first release) to describe the duration of your involvement.	
A8. In what city and country do you do most of your architecture work?	City: Country:
A9. For what company or organization do you work? If you can, give the name of the division or department if applicable.	
A10. Please list any quality methods your organization uses that apply to software. For each method, indicate the results of any evaluation of its use of the method (e.g., CMMI, independently assessed, staged Level 2; or ISO-9001, certified by an internal auditor).	

B. About your organization's architectural practices

B1. How important are the following activities for a software development organization to achieve high competence in software architecture? B2. How well does your organization carry out these activities now?	**How important are these activities to architecture competence?** **1 = Unimportant** **2 = Not very important** **3 = Fairly important** **4 = Very important** **? = Not sure**	**How well are these currently practiced by your organization?** **0 = not done at all** **1 = done poorly** **2 = done moderately well** **3 = done very well** **? = I don't know**
a. Hire talented architects.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
b. Establish a career track for architects.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
c. Make the position of architect highly regarded through visibility, reward, and prestige.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
d. Establish a clear statement of responsibilities and authority for architects.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
e. Establish a mentoring program for architects.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
f. Establish an architecture training and education program.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
g. Track how architects spend their time.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
h. Establish an architect certification program.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
i. Have architects receive external architect certifications.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
j. Measure architects' performance.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
k. Establish a forum for architects to communicate, and share information and experience.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
l. Establish a repository of reusable architectures and architecture-based artifacts.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
m. Develop reusable reference architectures.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
n. Establish organization-wide architecture practices.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
o. Establish an architecture review board.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐

p. Measure quality of architectures produced.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
q. Provide a centralized resource to analyze and help with architecture tools.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
r. Hold an organization-wide architecture conference.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
s. Initiate software process improvement or software quality improvement practices.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
t. Have architects join professional organizations.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
u. Bring in outside expert consultants on architecture.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
v. Include architecture milestones in project plans.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
w. Have architect provide input into product definition.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
x. Have architect advise on development team structure.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
y. Give architect influence throughout entire project life cycle.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
z. Reward/penalize architect based on project success.	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
aa. Other (specify):	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
bb. Other (specify):	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
cc. Other (specify):	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
dd. Other (specify):	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐
ee. Other (specify):	1☐ 2☐ 3☐ 4☐ ?☐	0☐ 1☐ 2☐ 3☐ ?☐

Notes, comments, or elaboration (optional):

C. About your duties, skills, and knowledge as a software architect

<table>
<tr><td rowspan="2">C1. What are your 5-8 most important duties as a software architect?
What percentage of your time does each one take? (The times need not total 100%.)</td><td>Duty</td><td>% time</td></tr>
<tr><td>a.
b.
c.
d.
e.
f.
g.
h.</td><td>a.
b.
c.
d.
e.
f.
g.
h.</td></tr>
<tr><td>C2. What are the 5-8 personal skills that you find most valuable as a software architect?</td><td colspan="2">a.
b.
c.
d.
e.
f.
g.
h.</td></tr>
<tr><td>C3. What are the 5-8 areas of technical knowledge that you find most valuable as a software architect?</td><td colspan="2">a.
b.
c.
d.
e.
f.
g.
h.</td></tr>
<tr><td colspan="3">C4. Ideally, what percent of time should a software architect spend actually working directly on the architecture?</td></tr>
<tr><td colspan="3">C5. In your role as software architect, what percentage of your time do you actually spend working directly on the architecture?</td></tr>
</table>

D. Assessing competence

D1. On a scale of 1 (very low) to 10 (very high), how would you rate the architecture competence of your organization?	
D2. On a scale of 1 (very low) to 10 (very high), how would your rate your own competence as a software architect?	
D3. How does your organization define or measure overall success?	
D4. On a scale of 1 (very low) to 10 (very high), rate your organization's overall success.	
D5. In your opinion, does your organization's work on software architecture have an effect on:	**-2 = strong negative effect** **-1 = somewhat negative effect** **0 = no effect** **1 = somewhat positive effect** **2 = strong positive effect** **? = don't know or not sure**
a. overall product quality	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
b. time to market or deployment	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
c. ability to derive new products from existing ones	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
d. cost of products	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
e. productivity of software development staff	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
f. level of software expertise of the technical staff	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
g. effective project planning	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
h. handling impact of changes to software and hardware	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
i. other (specify)	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
j. other (specify)	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
k. other (specify)	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐
l. other (specify)	-2☐ -1☐ 0☐ 1☐ 2☐ ?☐

Notes, comments, or elaboration (optional):

E. About this survey

E1. If we have questions about your responses, may we contact you?	☐ **Yes** ☐ **No**
E2. Would you be willing to participate in other surveys of about the same length related to architecture competence?	☐ **Yes** ☐ **No**
E3. If the answer to E1 or E2 is "yes," please enter your preferred email address.	
E4. Approximately how many minutes did it take you to complete the survey?	0-20 ☐ 21-30 ☐ 31-40 ☐ More than 40 ☐
E5. Is there anything else you'd like to tell us?	

Thank you very much for your participation. If you have questions or concerns about the survey, please send email to clements@sei.cmu.edu.

Please return this survey by email to clements@sei.cmu.edu.

Appendix B: Complete List of Duties, Skills, and Knowledge

The following tables contain the complete set of duties, skills, and knowledge collected during our information-gathering activity described in Section 2. The data is clustered into groups (lefthand column) and subgroups (middle column) as the result of an affinity exercise.

Table 7: Duties of a Software Architect

General Duty Area	Specific Duty Area	Duties
Architecting	**Creating an architecture**	Creating/designing an architecture. Choose an architecture. Create software architecture design plan. Based on an analysis of the given requirements, draw the initial architecture. Build product line architecture. Create architectural-level designs depicting the domain characteristics. Defining the product architecture. Make design decisions. Expand detail and refine design to converge on final design. Identifying the style and articulating the principles and key mechanisms of the architecture partitioning the system. Follow the scenarios patterns. Define how the various components fit together. Applies design patterns and uses UML tools.
	Architecture evaluation and analysis	Re-evaluate the architecture for implementing use cases & other requirements such as performance, scalability etc. Create prototypes. Participating in design reviews. Review construction-level designs. Review the Designs of the components designed by junior engineers. Reviewing designs for compliance with the architecture. Compare software architecture evaluation techniques. Apply value-based architecting techniques to evaluate architectural decisions. Modeling alternatives. Evaluating the architecture through various means including prototyping, reviews, and assessments. Analyzing and validating the architecture. Refine the architecture by integrating several principles. Identify the need and modify the technical architecture to accommodate project needs. Check requirements constraints, and refine the architecture accordingly. Review other people's architecture. Tradeoff analysis. Analyze styles product factors. Analysis of software architectural patterns. Experiment/simulate (analysis of the software under development). Analyze and implement architectural framework. Responsibility to take a series of views and a wider perspective on the system design.
	Documentation	Thoroughly understand and document the areas (domains) for which the system will be built. Prepare architectural documents and presentations. Document software interfaces. Produce a comprehensive documentation package for architecture useful to stakeholders. Keeping reader's point of view in mind while documenting. Creating, standardizing and using architectural descriptions. Use a certain documentation standard. Document variability and dynamism. Create conceptual architectural view. Create blueprints. Maintain architectural documentation.
	Existing system and transformation	Take care of existing system and its architecture. Understand the existing architectures & redesign it for migration to new technology & platforms. Transforming legacy architecture for present use. Reconstructing software architectures/reuse architecture.

Table 7: Duties of a Software Architect, cont'd.

General Duty Area	Specific Duty Area	Duties
Architecting (cont'd.)	**Other architecting duties not specific to the above categories**	Sell the vision, keep the vision alive. Participating in all product design meetings. Specialist technical advice on architecture, design n development. Provides architectural guidelines for all software design activities. Take a system viewpoint. Lead software process/architecture improvement activities. Identify and define the set of architectural scopes. Take overall responsibility of the architecture. Discover architecture principles that let the architecture meet its goals. Define philosophy and principles for global architecture. Overseeing and/or managing the architecture definition process. Focus on the big picture. Provide architecture oversight of software development projects. Using quality attributes.
Life-cycle phases other than architecture	**Requirements**	Analyze software requirements. Understanding business and customer needs. Capture customer, organizational and Business requirements on the architecture. Create software specifications from business requirements. Articulating and refining architectural requirements. Ensure that the requirements meet the company's needs. Listen to understand the scope of the project, the client's key design points, requirements, and expectations. Understanding the quality attributes. Assess the totality of the client's needs and resources before construction even begins. Document the defined requirements. Setting system scope (is/not; now/later). To set the functional path of how the system operates. The desired behaviors of the system are outlined. Getting input on needs to evolve and improve the architecture.
	Coding	Coding. Conducts code reviews. Develops reusable software components. Analyses, selects and integrates software components. Setting and ensuring adherence to coding guidelines. Recommending development methodologies and coding standards. Monitors, mentors and reviews the work of outside consultants and vendors. Handle the change-modifying of the code. Participation in the development of software components.
	Testing	Support field testing. Supports system testers. Responsible for bug-fixing and maintenance. Architecture-based testing. Conducts/supports components testing.
	Future technologies	Evaluates and recommends enterprise's software solutions. Manage the introduction of new software solutions. Analyze current IT environment and recommend solutions for deficiencies. Work with vendors to represent organization's requirements and influence future products. Develop and present technical white papers. Perform proof-of-concept of new software solutions / technologies.
	Tools and technology selection	Selecting key technologies. Technical feasibility studies of new technology and architecture. Identifies possible solutions through technology and organizational management & product changes. Specify required tools and methodologies. Leveraging information technology to best advantage. Tool selection: How to find the right tool for the enterprise. Maintains state-of-the-art knowledge of technologies, planning, design, and analysis methodologies. Lead discussions about technology standards. Evaluate commercial tools and software components from an architectural perspective. Review technical solutions at software level. Define development tools to be used. Technology trend analysis/roadmaps. Develop internal technical standards and contribute to the development of external technical standards.

Table 7: Duties of a Software Architect, cont'd.

General Duty Area	Specific Duty Area	Duties
Interacting with stakeholders	**Interacting with stakeholders in general or stakeholders other than clients or developers**	Take and retake the pulse of all critical influencers of the architecture project. Works with designers, technologists, and researchers to ensure user interface reflects client, user, and design requirements. Maintain medium to high level of visibility across the Product Group. Effective mediator between and among software developers, project management, and stakeholder. Identifying stakeholders and listening to and understanding what each wants. Describe to the stakeholders intended architecture through the stages of development. Convincing the stakeholders. Communicate, Listen, network, influence. Address the concerns of all the stakeholders. Choose the set of views that will be most valuable to the architecture's community of stakeholders. Interact with business experts to design state of the art systems. Assess the extent to which the architecture meets the various concerns of its stakeholders. Communicate critical decisions and their impact to the respective stakeholders.
	Clients	Create a plan along with the client articulating and communicating vision to the stakeholders. Be engaged with the client to ensure solution success. Guide the client through the construction process. Satisfy business objectives of the client. Convince the client to use the company's software. Preparing presentations for the clients. Customer interaction. Identifying the actual builders of the system. Take care of those who construct the system.
	Developers	Understand what the developers want and need from the architecture. Help developers see the value of the architecture and understand how to use it successfully. Gives them all the details needed. Guide the development team in implementing the system, and anticipate/diagnose problems, find/develop solutions to those problems. Create and teach tutorials to help developers.
Management	**Project management**	Budgeting and planning. Work in budgetary constraints. Sizing and estimation. Correlation between architecture design methods and project planning methods. Mapping architectures to implementations. Migration and risk assessment, managing risks. Problems and issues related/risk handling. Take care of configuration control. Ensures maintainable software. Create development schedules. Support the project by guiding and solving problems all along the execution cycle. Measure results using quantitative metrics and improve both personal results and teams' productivity. Achieving quality. Ensure software quality and conformance with industry and project standards. Provide support for applications. Handling resources is to make sure that they fully understand the economics. Responsible to bring in state-of-the art processes covering the development cycle. Plays a central role for the successful progression of projects. Improve software development practices. Identifying and scheduling architectural releases. Define and document construction process and sequence. Software architecture implementation.
	People management	Build "trusted advisor" relationships. Coordinates. Motivate. Advocates. Delegating or handing off. Managing resources. Acts as a supervisor.

Table 7: Duties of a Software Architect, cont'd.

General Duty Area	Specific Duty Area	Duties
Management (cont'd.)	**Support for management**	Provide feedback on appropriateness and difficulty of project. A good connection between software architecture and project management. Advise the project manager on the tradeoffs between software design choices and requirements choices. Provide input to Software Project Manager in the software project planning and estimation process. Serve as a "bridge" between the technical team and the PM/manager. Interacting with managers.
Organization and business related	**Organization**	Growing an architecture evaluation capability in your organization. Review and contribute to research and development efforts. Perform additional job-related duties as requested. Strategic role. Participate in the hiring process for the team. To give excellent value for the money to your employer. Product marketing. Institute and oversee cost-effective software architecture design reviews. Instrumental in developing intellectual property. Help the organization to invent and transform.
	Business	Translate business strategy into technical vision and strategy. Influencing business strategy. Understanding and evaluating business processes. Clearly understanding the business value of software architecture. Help the organization meet its business goals. Understand customer and market trends. Participate in automation of business processes. Identify, understanding and resolve business issues. Align architecture with the business goals & objectives.
Leadership and team building	**Technical leadership**	Lead a team of architects. Act as a technical leader. Making technical decisions. Decision making-deciding to build or to buy major parts of your system architecture. Make decisions.
	Team building	Build teams. Building the architecture team and aligning them behind the vision. Set team context (vision). Mentor junior architects. Consulting and educating the team on the use of the architecture. Maintain morale, both within and outside the architecture group. Foster the professional development of team members. Systematically checking your views to adopt a team methodology that suits your level of comfort. Coach teams of software design engineers for planning, tracking & completion of work within the agreed plan. Mentor and coach staff in the use of software technologies. Works both as a leader and an individual contributor.

Table 8: Skills of a Software Architect

General Skill Area	Specific Skill Area	Skills
Communication skills	**Outward**	Oral and written communication skills. Presentation skills. Can present and explain technical information to diverse audiences. Capable of transferring knowledge. Convincing skills.
	Communication skills in general	Communication skills. See from and sell to multiple viewpoints. Consulting skills. Negotiation skills. Can understand and express complex topics.
	Inward	Listening skills. Approachable. Interviewing.
Interpersonal skills	**Within team**	Team player. Work effectively with superiors, colleagues and customers. Collaborating. Maintains constructive working relationships. Work in a diverse team environment. Inspire creative collaboration. Consensus building. Balanced participation.
	With other people	Interpersonal skills. Diplomatic. Mentoring. Handles and resolves conflict. Should respect people. Committed to others' success.
Work skills	**Leadership**	Leadership. Decision making skills. Take initiative and is innovative. Self motivated and directed. Committed, dedicated, and passionate. Independent judgment. Influential. Ambitious. Commands respect. Charismatic.
	Effectively managing workload	Work well under pressure. Planning skills. Time management skills. Priority assessment. Result-oriented. Estimates well. Can support a wide range of issues. Can work on multiple complex projects concurrently. Effectively prioritize and execute tasks in a high-pressure environment. Can work on multiple complex systems concurrently.
	Skills to excel in corporate environment	Unflinching passion for quality. Art of Strategy. Work under general supervision. Work under given constraints. Organizational and workflow skills. Sensitive to where the power is and how it flows in your organization. Process-oriented. Political sagacity. Willingness to do what it takes to get the job done. Entrepreneurial. Assertive without being aggressive. Open to constructive criticism.
	Skills for handling information	Detail oriented while maintaining overall vision and focus. See a large picture. Good at working at an abstract level.
Personal skills	**Personal qualities**	Credible. Accountable. Responsible. Insightful. Visionary. Creative. Perseverant. Practical. Confident. Patient. Empathetic. Work ethics.
	Skills for handling unknown	Tolerant of ambiguity. Risk taking/managing skills. Problem solving skills. Reasoning. Analytical skills.
	Skills for handling unexpected	Adaptable. Flexible. Open minded. Resilient. Compromising.
	Learning	Learning. Good grasping power. Investigative. Observation power. Adept in using tools.

Table 9: Knowledge of a Software Architect

General Knowledge Area	Specific Knowledge Area	Specific Knowledge
Computer science knowledge	**Knowledge of architecture concepts**	Working knowledge of software architecture. Knowledge of architecture frame works. Knowledge of architectural patterns. Knowledge of standard architectures. Understanding of system architecture. Knowledge of architecting notations. Knowledge of Architecture Description Languages (ADL). Knowledge of software architecture view pts and styles. Knowledge of architecture description languages. Knowledge of emerging technologies like model-driven architecture. Know about Innovations and advances in software architecture. Knowledge of architecture evaluation models. Knowledge of all components of a technical architecture. Architecture techniques. Know about reliability, manageability, and maintainability, not to mention security. Knowledge of quality models. Software Architecture concepts.
	Knowledge of software engineering	Specialized knowledge of software engineering. Basic knowledge of software engineering. Knowledge of systems engineering. Knowledge of software development life cycle (SDLC). Software process management and improvement techniques. Knowledge of security, performance, design, maintenance issues. Strong abilities in requirements analysis. Strong mathematics background. Development methods and modeling techniques. Elicitation techniques. Consulting frameworks. Knowledge of component based software development. Principles of design and management of product line architectures. Reuse methods and techniques. Software product line techniques. Documentation experience. Knowledge of testing/debugging tools. Troubleshooting. Experience in testing. Product demonstrations and product installation experience. Experience of deploying successful software projects.
	Design knowledge	Technical Solution Design experience. Rooted in years of experience dealing with design issues that cut across a variety of technologies. Knowledge about different tools and design techniques. Experience of designing complex multi-product systems. Knowledge of object-oriented analysis and design (OOAD). UML diagrams and UML analysis modeling.
	Programming knowledge	Programming Knowledge/experience. Other programming language experience. Experience of developing and implementing middleware components. Knowledge of how robust software can be built and maintained. Implementation experience. Working knowledge of Java. Experience in technical development.
Knowledge of technologies and platforms	**Specific technologies and platforms**	Knowledge of computer software. Knowledge of hardware/software interfaces. Understanding of web-based applications. Experience with Web Services Technologies. Internet technologies. Knowledge of a specific software/operating system. Experience in Sun and Microsoft platforms. SQL and RDBMS Concepts/ Data Base experience.
	General knowledge of technologies and platforms	Strong technical breadth and depth. Knowledge of technical analysis/evaluation techniques. Know benefits from having knowledge of the technologies. Knowledge about IT industry future directions. Familiarity with the ways in which infrastructure impact an application. Know what technical issues are key to success. Needs to have good technical judgment. Knowledge about the previous technologies.

Table 9: Knowledge of a Software Architect (cont.'d)

General Knowledge Area	Specific Knowledge Area	Specific Knowledge
Knowledge about the organization's context and management	**Domain knowledge**	Knowledge of different domains. In-depth understanding of the domain and pertinent technologies. Specific domain experience (as needed by the company). Security domain experience. Experience handling projects involving large volumes of data. Experience with real-time systems, video systems. Knowledge of data warehousing.
	Industry knowledge	Knowledge of industry's best practices. Familiarity with Industry standards. Experience working in onshore/offshore team environment. Should be a specialist.
	Enterprise knowledge	Knowledge about the company's business practices. Your competition (products, strategies and processes). A sound sense of business and technical strategy. Business re-engineering principles and processes. Organization's business strategy and rationale. What key players want, both business and personal. Knowledge of customer segment. Sales and/or customer expertise. Strong competitive experience. Customer training experience. Know all stakeholders viewpoints and their perspectives. Strategic planning experience. Knowledge of financial models and budgeting.
	Leadership and management techniques and experience	Yourself. Leadership experience. Managerial experience. History of coaching, mentoring and training software developers. Knowledge of project management. Knowledge of project engineering.

Bibliography

URLs are valid as of the publication date of this document.

[America 2003]
America, P.; Obbink, H.; & Rommes, E. "Multi-View Variation Modeling for Scenario Analysis." *Proceedings of the Fifth International Workshop on Product Family Engineering* (PFE-5). Sienna, Italy, 2003, Springer-Verlag, pp. 44-65.

[American Heritage 2002]
American Heritage Editors. *The American Heritage Stedman's Medical Dictionary.* Houghton-Mifflin, 2002.

[Argote 2007]
Argote, L. & Todorova, G. *International Review of Industrial and Organizational Psychology.* John Wiley & Sons, Ltd, 2007 (ISBN: 978-0-470-03198-8).
http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470031980,descCd-tableOfContents.html

[Batman 2008]
Batman, Joe. *Hunting the Product Line Architecture.*
http://www.sei.cmu.edu/architecture/batman.pdf (2008).

[Beyer 1998]
Beyer, H. & Holzblatt, K. *Contextual Design: Defining Customer-Centered Systems*. ACM Press, 1998 (ISBN 1-558-60411-1). As cited in Bass, L.; Nord, R.; Wood, W.; & Zubrow, D. *Risk Themes Discovered Through Architecture* (CMU/SEI-2006-TR-012, ADA456884). Software Engineering Institute, Carnegie Mellon University, 2006.
http://www.sei.cmu.edu/publications/documents/06.reports/06tr012.html

[Binder 2006]
Binder Riha Associates. *The Six Boxes*. http://www.sixboxes.com/The_Model.html (2006).

[Bloom 2004]
Bloom, Benjamin. *Bloom's Taxonomy,* Adapted from Benjamin S. Bloom "Taxonomy of Educational Objectives." Allyn and Bacon, copyright Pearson Education, 1984.
http://www.coun.uvic.ca/learn/program/hndouts/bloom.html (2004).

[Boehm 2007]
Boehm, B.; Valerdi, R.; & Honour, E. "The ROI of Systems Engineering: Some Quantitative Results." *Proceedings of Seventeenth International Symposium of the International Council on Systems Engineering (INCOSE)*. San Diego, CA, June 2007. INCOSE, 2007.
http://www.incose.org/symp2007/

[Bredemeyer 2007]
Bredemeyer, Dana. *About Dana Bredemeyer*.
http://www.bredemeyer.com/DanaBredemeyer.htm (2007).

[Cataldo 2007]
Cataldo, Marcelo; Bass, Matthew; Herbsleb, James D.; & Bass, Len. "On Coordination Mechanisms in Global Software Development," 71-80. *International Conference on Global Software Engineering (ICGSE 2007)*. Munich, Germany, Aug. 2007. IEEE, 2007.

[Clements 2007]
Clements, Paul; Kazman, Rick; Klein, Mark; Devesh, Divya; Shivani, Reddy; & Verma, Prageti. "The Duties, Skills, and Knowledge of Software Architects." *Sixth Working IEEE/IFIP Conference on Software Architecture (WICSA 2007)*. Mumbai, India, Jan. 2007. IEEE, 2007.
http://www.gv.psu.edu/WICSA2007/tech.htm

[Dutton 1984]
Dutton, J. M. & Thomas, A. "Treating Progress Functions as a Managerial Opportunity." *Academy of Management Review 9* (1984): 235-247.

[Fiol 1985]
Fiol, C. M. & Lyles, M. A. "Organizational Learning." *Academy of Management Review 10,* 4 (1985): 803.

[Garcia 2007]
Garcia, Suzanne & Turner, Richard. *CMMI Survival Guide: Just Enough Process Improvement*. Addison-Wesley, 2007 (ISBN: 0321422775).
http://www.sei.cmu.edu/publications/books/process/cmmi-survival-guide.html

[Gilbert 1978]
Gilbert, Thomas F. *Human Competence-Engineering Worthy Performance: Engineering Worthy Performance.* McGraw-Hill, 1978.

[Gilbert 1996]
Gilbert, Thomas F. *Human Competence-Engineering Worthy Performance: Engineering Worthy Performance*. International Society for Performance Improvement, 1996 (ISBN: 0961669012, 978-0961669010).

[Hamel 1990]
Hamel, Gary & Prahalad, C. K. *The Core Competence of the Corporation (HBR OnPoint Enhanced Edition)*. Harvard Business Review, 1990.

[Hoffman 1999]
Hoffman, Terrence. "The Meanings of Competency." *Journal of European Industrial Training 23,* 6 (1999): 275-286.

[INCOSE 2005]
International Council on Systems Engineering. *Engineering Core Competencies Framework,* 2005.
http://www.incose.org.uk/Downloads/Core%20Competencies%2021-July-05.pdf

[Jeffcoat 2005]
Jeffcoat, Sandra Y. & Yaghoobi, Ahmad R. "System Architect: Necessity, Not Luxury." *Perspectives of the IASA 2,* 1 (August 2005): 8-10.

[Kaplan 1996]
Kaplan, Robert S. & Norton, David P. *The Balanced Scorecard: Translating Strategy into Action.* Harvard Business School Press, 1996 (ISBN: 0875846513, 978-0875846514).

[Kazman 2005]
Kazman, Rick & Bass, Len. *Categorizing Business Goals for Software Architectures* (CMU/SEI-2005-TR-021, ADA444917). Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr021.html

[Levitt 1988]
Levitt, Barbara & March, James G. "Organizational Learning." *Annual Review of Sociology 14,* 1 (1988): 319-40.

[Lewis 2001]
Lewis, Michael A. "Success, Failure and Organizational Competence. A Case Study of the New Product Development Process." *Journal of Engineering and Technology Management 18,* 1 (2001): 185–206.

[Luftman 1993]
Luftman, J.; Lewis, P.; & Oldach, S. "Transforming the Enterprise: The Alignment of Business and Information Technology Strategies." *IBM Systems Journal 32,* 1 (1993): 198-221.

[McGregor 1967]
McGregor, Douglas. *The Professional Manager*. New York, NY: McGraw-Hill Education, 1967 (ISBN: 0070450935, 978-0070450936).

[Microsoft 2008]
Microsoft, Inc. *Architecture Resource Center.*
http://www.microsoft.com/architecture/default.aspx?pid=share.certification (2008).

[Nickols 2007]
Nickols, Fred. "McGregor Meets Gilbert." http://home.att.net/~essays/mcgregorgilbert.pdf (2007).

[Pritchard 1987]
Pritchard, James. "From Shipwright to Naval Constructor: The Professionalization of 18th Century French Naval Shipbuilders." *Technology and Culture 28,* 1 (January 1987).

[Random House 2006]
Dictionary.com. *Dictionary.com Unabridged* (V1.0.1), Based on the *Random House Unabridged Dictionary*. Random House, Inc., 2006. http://dictionary.reference.com/browse

[SEI 2008]
Software Engineering Institute. "Software Architecture for Software-Intensive Systems." http://www.sei.cmu.edu/architecture/arch_duties.html (2008).

[Shaw 2006]
Shaw, Mary & Clements, Paul. *The Golden Age of Software Architecture: A Comprehensive Survey* (CMU-ISRI-06-101). School of Computer Science, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/architecture/GoldenAgeTR_v6.pdf

[Taatila 2004]
Taatila, Vesa. *The Concept of Organizational Competence - A Foundational Analysis*. University of Jyvaskyla, 2004. http://dissertations.jyu.fi/studcomp/9513917185.pdf

[Tague 2005]
Tague, Nancy R. *The Quality Toolbox*. ASQ Quality Press, 2005 (ISBN: 0873896394). http://www.loc.gov/catdir/toc/ecip055/2004029947.html

[Teodorescu 2004]
Teodorescu, Tina M. & Binder, Carl. *Competence Is What Matters*. International Society for Performance Improvement (ISPI), 2004.
http://www.sixboxes.com/Articles/Teodorescu_Binder.pdf

[Turley 1995]
Turley, Richard T. & Bieman, James M. "Competencies of Exceptional and Non-Exceptional Software Engineers." *Journal of Systems and Software 28*, 1 (Jan. 1995): 19-38.

[van Ommering 2005]
van Ommering, Rob. "Things to Do in Denver if You're an Architect." 2005.
http://www.sei.cmu.edu/architecture/ThingsToDoInDenver.htm

[Webster 1996]
Webster New World Dictionary. *Webster's New Universal Unabridged Dictionary*. Barnes & Noble Books, 1996.

[WICSA 2007]
WICSA Conference Wiki. *Working IFIP/IEEE Conference on Software Architecture Session: Education--Discussions*.
http://wwwp.dnsalias.org/wiki/Session:Education--Discussions (2007).

[Wikipedia 2007]
Wikipedia. "Thomas. F. Gilbert Biography." 2007.
http://en.wikipedia.org/wiki/Thomas_F_Gilbert

[Wikipedia 2008]
Wikipedia. “Dependency Structure Matrix.” 2008.
http://en.wikipedia.org/wiki/Dependency_Structure_Matrix

[Woodruffe 1993]
Woodruffe, Charles “What Is Meant by a Competency?” *Leadership & Organization Development Journal 14,* 1 (1993): 29.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. **AGENCY USE ONLY**
(Leave Blank)

2. **REPORT DATE**
March 2008

3. **REPORT TYPE AND DATES COVERED**
Final

4. **TITLE AND SUBTITLE**
Models for Evaluating and Improving Architecture Competence

5. **FUNDING NUMBERS**
FA8721-05-C-0003

6. **AUTHOR(S)**
Len Bass, Paul Clements, Rick Kazman, Mark Klein

7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. **PERFORMING ORGANIZATION REPORT NUMBER**
CMU/SEI-2008-TR-006

9. **SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. **SPONSORING/MONITORING AGENCY REPORT NUMBER**
ESC-TR-2008-006

11. **SUPPLEMENTARY NOTES**

12A **DISTRIBUTION/AVAILABILITY STATEMENT**
Unclassified/Unlimited, DTIC, NTIS

12B **DISTRIBUTION CODE**

13. **ABSTRACT (MAXIMUM 200 WORDS)**

Software architecture competence is the ability of an individual or organization to acquire, use, and sustain the skills and knowledge necessary to carry out software architecture-centric practices. Previous work in architecture has concentrated on its technical aspects: methods and tools for creating, analyzing, and using architecture. However, a different perspective recognizes that these activities are carried out by people working in organizations, and those people and organizations can use assistance towards consistently producing high-quality architectures.

This report lays out the basic concepts of software architecture competence and describes four models for explaining, measuring, and improving the architecture competence of an individual or a software-producing organization. The models are based on (1) the duties, skills, and knowledge required of a software architect or architecture organization, (2) human performance technology, an engineering approach applied to improving the competence of individuals, (3) organizational coordination, the study of how people and units in an organization share information, and (4) organizational learning, an approach to how organizations acquire, internalize, and utilize knowledge to improve their performance. The report also shows how the four models can be synergistically applied to produce an evaluation instrument to measure an organization's architecture competence.

14. **SUBJECT TERMS**
architecture competence model, DSK model, Duties Skills Knowledge model, Organizational Learning model, Human Performance Technology model, Organizational Coordination model, survey of practicing architects

15. **NUMBER OF PAGES**
83

16. **PRICE CODE**

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL